

**Université de Montréal**

**Learning visual representations with neural networks  
for video captioning and image generation**

par

**Li Yao**

Département d' informatique et de recherche opérationnelle  
Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures  
en vue de l'obtention du grade de  
Philosophiæ Doctor (Ph.D.)  
en informatiques

décembre 2017



# RÉSUMÉ

---

La recherche sur les réseaux de neurones a permis de réaliser de larges progrès durant la dernière décennie. Non seulement les réseaux de neurones ont été appliqués avec succès pour résoudre des problèmes de plus en plus complexes; mais ils sont aussi devenus l'approche dominante dans les domaines où ils ont été testés tels que la compréhension du langage, les agents jouant à des jeux de manière automatique ou encore la vision par ordinateur, grâce à leurs capacités calculatoires et leurs efficacités statistiques.

La présente thèse étudie les réseaux de neurones appliqués à des problèmes en vision par ordinateur, où les représentations sémantiques abstraites jouent un rôle fondamental. Nous démontrerons, à la fois par la théorie et par l'expérimentation, la capacité des réseaux de neurones à apprendre de telles représentations à partir de données, avec ou sans supervision.

Le contenu de la thèse est divisé en deux parties. La première partie étudie les réseaux de neurones appliqués à la description de vidéo en langage naturel, nécessitant l'apprentissage de représentation visuelle. Le premier modèle proposé permet d'avoir une attention dynamique sur les différentes trames de la vidéo lors de la génération de la description textuelle pour de courtes vidéos. Ce modèle est ensuite amélioré par l'introduction d'une opération de convolution récurrente. Par la suite, la dernière section de cette partie identifie un problème fondamental dans la description de vidéo en langage naturel et propose un nouveau type de métrique d'évaluation qui peut être utilisé empiriquement comme un oracle afin d'analyser les performances de modèles concernant cette tâche.

La deuxième partie se concentre sur l'apprentissage non-supervisé et étudie une famille de modèles capables de générer des images. En particulier, l'accent est mis sur les "Neural Autoregressive Density Estimators (NADEs)", une famille de modèles probabilistes pour les images naturelles. Ce travail met tout d'abord en évidence une connection entre les modèles NADEs et les réseaux stochastiques génératifs (GSN). De plus, une amélioration des modèles NADEs standards est proposée. Dénommés NADEs itératifs, cette amélioration introduit plusieurs itérations lors de l'inférence du modèle NADEs tout en préservant son nombre de paramètres.

Débutant par une revue chronologique, ce travail se termine par un résumé des récents développements en lien avec les contributions présentées dans les deux parties principales,

concernant les problèmes d'apprentissage de représentation sémantiques pour les images et les vidéos. De prometteuses directions de recherche sont envisagées.

**Mot clés:** réseaux de neurones, apprentissage de représentations, description naturelle de vidéos, apprentissage supervisé, apprentissage non-supervisé, représentation visuelle

# ABSTRACT

---

The past decade has been marked as a golden era of neural network research. Not only have neural networks been successfully applied to solve more and more challenging real-world problems, but also they have become the dominant approach in many of the places where they have been tested. These places include, for instance, language understanding, game playing, and computer vision, thanks to neural networks' superiority in computational efficiency and statistical capacity.

This thesis applies neural networks to problems in computer vision where high-level and semantically meaningful representations play a fundamental role. It demonstrates both in theory and in experiment the ability to learn such representations from data with and without supervision.

The main content of the thesis is divided into two parts. The first part studies neural networks in the context of learning visual representations for the task of video captioning. Models are developed to dynamically focus on different frames while generating a natural language description of a short video. Such a model is further improved by recurrent convolutional operations. The end of this part identifies fundamental challenges in video captioning and proposes a new type of evaluation metric that may be used experimentally as an oracle to benchmark performance.

The second part studies the family of models that generate images. While the first part is supervised, this part is unsupervised. The focus of it is the popular family of Neural Autoregressive Density Estimators (NADEs), a tractable probabilistic model for natural images. This work first makes a connection between NADEs and Generative Stochastic Networks (GSNs). The standard NADE is improved by introducing multiple iterations in its inference without increasing the number of parameters, which is dubbed iterative NADE.

With a historical view at the beginning, this work ends with a summary of recent development for work discussed in the first two parts around the central topic of learning visual representations for images and videos. A bright future is envisioned at the end.

**keywords:** neural networks, representation learning, video captioning, unsupervised learning, supervised learning, visual representation



# CONTENTS

---

<b>Résumé</b> .....	i
<b>Abstract</b> .....	iii
<b>List of figures</b> .....	xi
<b>List of tables</b> .....	xv
<b>Acknowledgement</b> .....	xvii
<b>Chapter 1. Machine learning</b> .....	1
1.1. Machine learning and its relation with other subjects .....	1
1.2. Formalizing learning .....	2
1.2.1. Basic idea of Bayesian learning .....	3
1.2.2. Frequentist learning .....	3
1.2.3. Parametric and non-parametric learning .....	5
1.3. Learning to generalize .....	6
1.3.1. Generalization in supervised learning .....	6
1.3.2. Generalization in unsupervised learning .....	7
1.3.3. Formalizing the generalization error .....	7
1.3.4. Model selection .....	7
1.3.4.1. Bayesian model averaging .....	8
1.3.4.2. Cross-validation .....	8
1.3.4.3. Bagging and boosting .....	8
1.4. The curse of dimensionality and its implication .....	9
<b>Chapter 2. Representation learning</b> .....	11
2.1. Challenging problems in the AI-set .....	11
2.2. Learning hierarchical and distributed representations .....	12
2.3. Feedforward neural networks .....	13

2.4. Recurrent neural networks .....	15
2.5. Auto-encoders .....	16
2.6. Graphical models based neural networks .....	17
2.6.1. Directed graphical models .....	18
2.6.2. Variational autoencoders .....	20
2.6.3. Markov random fields .....	20
2.7. Computational techniques .....	23
2.7.1. Optimization .....	23
2.7.2. Sampling in graphical models .....	26
2.7.2.1. Markov chain Monte Carlo .....	26
<b>Chapter 3. Learning visual representations .....</b>	<b>29</b>
3.1. The computer vision challenge .....	29
3.2. visual representations .....	29
3.3. Traditional approaches .....	31
3.3.1. Image descriptors and encoding .....	31
3.3.2. Video descriptors and encoding .....	33
3.4. Learning-based approaches .....	33
3.4.1. Supervised learning of visual representation .....	33
3.4.2. Unsupervised learning of visual representation .....	34
<b>Chapter 4. Prologue to first article .....</b>	<b>37</b>
4.1. Article Detail .....	37
4.2. Context .....	37
4.3. Contributions .....	38
<b>Chapter 5. Describing Videos by Exploiting Temporal Structure .....</b>	<b>39</b>
5.1. Introduction .....	39
5.2. Video Description Generation Using an Encoder–Decoder Framework .....	41
5.2.1. Encoder-Decoder Framework .....	41
5.2.2. Encoder: Convolutional Neural Network .....	42
5.2.3. Decoder: Long Short-Term Memory Network .....	43



5.3. Exploiting Temporal Structure in Video Description Generation.....	44
5.3.1. Exploiting Local Structure: A Spatio-Temporal Convolutional Neural Net.....	44
5.3.2. Exploiting Global Structure: A Temporal Attention Mechanism.....	46
5.4. Related Work.....	47
5.5. Experiments.....	48
5.5.1. Datasets.....	48
5.5.1.1. Youtube2Text.....	48
5.5.1.2. DVS.....	49
5.5.2. Description Preprocessing.....	49
5.5.3. Video Preprocessing.....	49
5.5.4. Experimental Setup.....	50
5.5.4.1. Models.....	50
5.5.4.2. Training.....	51
5.5.4.3. Evaluation.....	51
5.5.5. Quantitative Analysis.....	51
5.5.6. Qualitative Analysis.....	52
5.6. Conclusion.....	53
Acknowledgments.....	53
<b>Chapter 6. Prologue to second article.....</b>	<b>55</b>
6.1. Article Detail.....	55
6.2. Context.....	55
6.3. Contributions.....	56
<b>Chapter 7. Oracle Performance for Visual Captioning.....</b>	<b>57</b>
7.1. Introduction.....	57
7.2. Related work.....	58
7.2.1. Visual captioning.....	58
7.2.2. Capturing higher-level visual concept.....	59
7.3. Oracle Model.....	59
7.3.1. Oracle Parameterization.....	60

7.3.2. Atoms Construction .....	61
7.4. Contributing factors of the oracle .....	61
7.5. Experiments .....	62
7.5.1. Atom extraction .....	62
7.5.2. Training .....	64
7.5.3. Interpretation .....	64
7.5.3.1. comparing oracle performance with existing models .....	64
7.5.3.2. quantify the diminishing return .....	64
7.5.3.3. atom accuracy versus atom quantity .....	66
7.5.3.4. intrinsic difficulties of particular datasets .....	66
7.6. Discussion .....	67
Acknowledgments .....	68
<b>Chapter 8. Prologue to third article .....</b>	<b>69</b>
8.1. Article Detail .....	69
8.2. Context .....	69
8.3. Contributions .....	70
<b>Chapter 9. Delving Deeper into Convolutional Networks for Learning Video Representations .....</b>	<b>71</b>
9.1. Introduction .....	71
9.2. GRU: Gated Recurrent Unit Networks .....	73
9.3. Delving Deeper into Convolutional Neural Networks .....	73
9.3.1. GRU-RCN .....	74
9.3.2. Stacked GRU-RCN: .....	75
9.4. Related Work .....	76
9.5. Experimentation .....	76
9.5.1. Action Recognition .....	77
9.5.1.1. Model Architecture .....	77
9.5.1.2. Model Training and Evaluation .....	78
9.5.1.3. Results .....	78
9.5.2. Video Captioning .....	80

9.5.2.1. Model Specifications .....	80
9.5.2.2. Training .....	80
9.5.2.3. Results .....	81
9.6. Conclusion .....	82
Acknowledgments .....	82
<b>Chapter 10. Prologue to fourth article .....</b>	<b>83</b>
10.1. Article Detail .....	83
10.2. Context .....	83
10.3. Contributions .....	84
<b>Chapter 11. On the Equivalence Between Deep NADE and Generative Stochastic Network .....</b>	<b>85</b>
11.1. Introduction .....	85
11.2. Deep NADE and Order-Agnostic Training .....	86
11.2.1. NADE .....	87
11.2.2. Deep NADE .....	87
11.3. Generative Stochastic Networks .....	88
11.4. Equivalence between deep NADE and GSN .....	90
11.4.1. Alternative Sampling Method for NADE .....	91
11.4.2. The GSN Chain Averages an Ensemble of Density Estimators .....	92
11.5. Annealed GSN Sampling .....	92
11.6. Experiments .....	94
11.6.1. Settings: Dataset and Model .....	94
11.6.2. Qualitative Analysis .....	95
11.6.3. Quantitative Results .....	95
11.7. Conclusions .....	97
Acknowledgments .....	98
<b>Chapter 12. Prologue to fifth article .....</b>	<b>99</b>
12.1. Article Detail .....	99

12.2.	Context .....	99
12.3.	Contributions .....	100
<b>Chapter 13.</b>	<b>Iterative Neural Autoregressive Distribution Estimator</b>	
	<b>(NADE-<math>k</math>) .....</b>	<b>101</b>
13.1.	Introduction .....	101
13.2.	Proposed Method: NADE- $k$ .....	102
13.2.1.	Related Methods and Approaches .....	104
13.3.	Experiments .....	106
13.3.1.	MNIST .....	106
13.3.1.1.	Qualitative Analysis .....	109
13.3.1.2.	Variability over Orderings .....	109
13.3.2.	Caltech-101 silhouettes .....	109
13.4.	Conclusions and Discussion .....	110
	Acknowledgements .....	112
<b>Chapter 14.</b>	<b>Recent development in learning visual representations...</b>	<b>113</b>
14.1.	Learning image representations .....	113
14.2.	Learning video representations .....	114
<b>Chapter 15.</b>	<b>Conclusion .....</b>	<b>117</b>
<b>Bibliography</b> .....		<b>119</b>

# LIST OF FIGURES

---

2.1	<i>Left:</i> A feed-forward neural network with one input layer, one hidden layer, and one output layer. <i>Right:</i> A feed-forward neural network in a general form whose organization is not according to layers.....	14
2.2	<i>Left:</i> An RNN with one recurrent layer, one input layer and one output layer. <i>Right:</i> An RNN unfolds itself over time steps to form a feed-forward model with shared hidden layers <i>The figure is credited by LeCun et al. (2015)</i> .....	15
2.3	<i>Left:</i> A standard auto-encoder must have an information bottleneck in the hidden layer to avoid learning the trivial identity reconstruction function. <i>Right:</i> A regularized auto-encoder that can have an over-complete hidden layer, such as denoising auto-encoders, and contractive auto-encoders since it cannot learn the identity function.....	16
2.4	<i>Left:</i> An example of directed graphical models. Edges are directed. The number in the nodes indicates a topological order. <i>Right:</i> The Restricted Boltzmann machine as an example of undirected graphical models. Edges are undirected.....	18
2.5	<i>Left:</i> SGD uses noisy gradient to reach global minima. <i>Right:</i> Second order optimization methods use a local quadratic approximation and move to the minimum of a quadratic curve at each step.....	24
3.1	A good representation is invariant to the physical interactions and environmental changes such as (a) – (e). A good representation may not be easily defined with handwritten rules such as (f) and (g). .....	30
3.2	Credit: <a href="http://www.learnopencv.com/">http://www.learnopencv.com/</a> . Classical computer vision approach may be separated into several stages each of which has its particular set of algorithms.....	31
3.3	Credit: (Tsai, 2012). Bag of visual words representation for images. Region-based descriptors are first clustered by $k$ -means in (iii), followed by a histogram	

	where each bin is a cluster, the size of the bin being the number of regional descriptors that fall within that cluster.....	32
5.1	High-level visualization of our approach to video description generation. We incorporate models of both the local temporal dynamic (i.e. within blocks of a few frames) of videos, as well as their global temporal structure. The local structure is modeled using the temporal feature maps of a 3-D CNN, while a temporal attention mechanism is used to combine information across the entire video. For each generated word, the model can focus on different temporal regions in the video. For simplicity, we highlight only the region having the maximum attention above. ....	40
5.2	Illustration of the spatio-temporal convolutional neural network (3-D CNN). This network is trained for activity recognition. Then, only the convolutional layers are involved when generating video descriptions. ....	45
5.3	Illustration of the proposed temporal attention mechanism in the LSTM decoder .....	46
5.4	Four sample videos and their corresponding generated and ground-truth descriptions from Youtube2Text (Left Column) and DVS (Right Column). The bar plot under each frame corresponds to the attention weight $\alpha_i^t$ for the frame when the corresponding word (color-coded) was generated. From the top left panel, we can see that when the word “road” is about to be generated, the model focuses highly on the third frame where the road is clearly visible. Similarly, on the bottom left panel, we can see that the model attends to the second frame when it was about to generate the word “Someone”. The bottom row includes alternate descriptions generated by the other model variations.	52
7.1	Given ground truth captions, three categories of visual atoms (entity, action and attribute) are automatically extracted using NLP Parser. “NA” denotes the empty atom set.....	63
7.2	Learned oracle on COCO (left), YouTube2Text (middle) and LSMDC (right). The number of atoms $\mathbf{a}^{(k)}$ is varied on x-axis and oracles are computed on y-axis on testsets. The first row shows the oracles on BLEU and METEOR with $3k$ atoms, $k$ from each of the three categories. The second row shows the oracles when $k$ atoms are selected individually for each category. CIDEr is used for COCO and YouTube2Text as each test example is associated	

	with multiple ground truth captions, argued in (Vedantam et al., 2014). For LSMDC, METEOR is used, as argued by Rohrbach et al. (2015).....	65
7.3	Learned oracles with different atom precision ( $r$ in red) and atom quantity (x-axis) on COCO (left) and LSMDC (right). The number of atoms $\hat{\mathbf{a}}_r^{(k)}$ is varied on x-axis and oracles are computed on y-axis on testsets. CIDEr is used for COCO and METEOR for LSMDC. It shows one could increase the score by either improving $P(\mathbf{a}^{(k)} \mathbf{v})$ with a fixed $k$ or increase $k$ . It also shows the maximal error bearable for different score. ....	67
9.1	Visualization of convolutional maps on successive frames in video. As we go up in the CNN hierarchy, we observe that the convolutional maps are more stable over time, and thus discard variation over short temporal windows....	72
9.2	High-level visualization of our model. Our approach leverages convolutional maps from different layers of a pretrained-convnet. Each map is given as input to a convolutional GRU-RNN (hence GRU-RCN) at different time-step. Bottom-up connections may be optionally added between RCN layers to form Stack-GRU-RCN. ....	74
11.1	Independent samples generated by the ancestral sampling procedure from the deep NADE. ....	93
11.2	The consecutive samples from the two independent GSN sampling chains without any annealing strategy. Both chains started from uniformly random configurations. Note the few spurious samples which can be avoided with the annealing strategy (see Figure 11.3). ....	94
11.3	Samples generated by the annealed GSN sampling procedure for the same deep NADE model. Visually the quality is comparable to the ancestral samples, and mixing is very fast. This is obtained with $p_{\max} = 0.9$ , $p_{\min} = 0.1$ , $\alpha = 0.7$ and $T = 20$ . ....	96
11.4	The annealed GSN sampling procedure is compared against NADE ancestral sampling, trading off the computational cost (computational wrt to ancestral sampling on x-axis) against log-likelihood of the generated samples (y-axis). The computational cost discards the effect of the Markov chain autocorrelation by estimating the effective number of samples and increasing the computational cost accordingly. ....	97

13.1	The choice of a structure for NADE- $k$ is very flexible. The dark filled halves indicate that a part of the input is observed and fixed to the observed values during the iterations. Left: Basic structure corresponding to Equations (13.2.6–13.2.7) with $n = 2$ and $k = 2$ . Middle: Depth added as in NADE by Uria et al. (2014) with $n = 3$ and $k = 2$ . Right: Depth added as in Multi-Prediction Deep Boltzmann Machine by Goodfellow et al. (2013) with $n = 2$ and $k = 3$ . The first two structures are used in the experiments. ....	103
13.2	The inner working mechanism of NADE- $k$ . The left most column shows the data vectors $\mathbf{x}$ , the second column shows their masked version and the subsequent columns show the reconstructions $\mathbf{v}^{(0)} \dots \mathbf{v}^{(10)}$ (See Eq. (13.2.7)).	104
13.3	NADE- $k$ with $k$ steps of variational inference helps to reduce the training cost (a) and to generalize better (b). NADE-mask performs better than NADE-1 without masks both in training and test.....	108
13.4	(a) The generalization performance of different NADE- $k$ models trained with different $k$ . (b) The generalization performance of NADE-5 2h, trained with $k=5$ , but with various $k$ in test time. ....	108
13.5	Samples generated from NADE- $k$ trained on (a) MNIST and (b) Caltech-101 Silhouettes.....	110
13.6	Filters learned from NADE-5 2HL. (a) A random subset of the encoding filters. (b) A random subset of the decoding filters. ....	110



# LIST OF TABLES

---

5. I	Performance of different variants of the model on the Youtube2Text .....	50
5. II	Performance of different variants of the model on the DVS.....	50
7. I	Human evaluation of proportion of atoms that are voted as visual. It is clear that extracted atoms from three categories contain dominant amount of visual elements, hence verifying the procedure described in Section 7.3.2. Another observation is that entities and actions tend to be more visual than attributes according to human perception.....	63
7. II	Measure semantic capacity of current state-of-the-art models. One could easily map the reported metric to the number of visual atoms captured. This establishes an equivalence between a model, the proposed oracle and a model’s semantic capacity. (“ENT” for entities. “ACT” for actions. “ATT” for attributes. “ALL” for all three categories combined. “B1” for BLEU-1, “B4” for BLEU-4. “M” for METEOR. “C” for CIDEr. Note that the CIDEr is between 0 and 10. The learned oracle is denoted in <b>bold</b> . ....	66
9. I	Classification accuracy of different variants of the model on the UCF101 split 1. We report the performance of previous works that learn representation using only RGB information only. ....	79
9. II	Performance of different variants of the model on YouTube2Text for video captioning. Representations obtained with the proposed RCN architecture combined with decoders from Yao et al. (2015a) offer a significant performance boost, reaching the performance of the other state-of-the-art models.....	81
11. I	Log-probability of 1000 samples when anealing is not used. To collect samples, 100 parallel chains are run and 10 samples are taken from each chain and combined together. The noise level is fixed at a particular level during the sampling. We also report the best log-probability of samples generated with an annealed GSN sampling. ....	96

13. I	Results obtained on MNIST using various models and number of hidden layers (1HL or 2HL). “Ords” is short for “orderings”. These are the average log-probabilities of the test set. EoNADE refers to the ensemble probability (See Eq. (13.2.9)). From here on, in all figures and tables we use “HL” to denote the number of hidden layers and “h” for the number of hidden units.	107
13. II	The variance of $\log p(\mathbf{x} \mid o)$ over orderings $o$ and over test samples $\mathbf{x}$ . . . . .	109
13. III	Average log-probabilities of test samples of Caltech-101 Silhouettes. (★) The results are from <a href="#">Cho et al. (2013)</a> . The terms in the parenthesis indicate the number of hidden units, the total number of parameters (M for million), and the L2 regularization coefficient. NADE-mask 670h achieves the best performance without any regularizations. . . . .	111

# ACKNOWLEDGEMENT

---

First of all, I would like to thank Prof. Yoshua Bengio for being a rock-star teacher and supervisor. I clearly remember my early days of graduate school when we sat down together debugging codes. My favorite quote from him is “Being a good researcher is like being a detective who does not overlook any suspicion in the code and in the experiments”. I remember the day when we got our first successful experiment for the NIPS paper in 2013 and he was so excited and we did Hi-Five three times. What I have learned from Prof. Bengio is that a scientist lives and breathes science, which I still find it hard to follow from time to time. All my work on unsupervised learning in this thesis were done with Prof. Bengio. He taught me to see models from a probabilistic point of view, be organized, patient and persevere in debugging and analyzing experiments.

I would also like to thank many people who have taught me so much in the past 5 years. Prof. Aaron Courville have worked with me closely on video projects and supported me through those sunny and rainy days. Prof. Kyunghyun Cho, who has been my co-author for almost all of my publications taught me the importance of scientific writing and his brilliance has been my inspiration for all these years. Dr. Nicolas Ballas spent most of his post-doc doing research with me, with whom I had my most productive period. In addition, I would like to thank people who have helped me with my research in various ways: Prof. Hugo Larochelle from Google, Dr. Razvan Pascanu, Dr. Guillaume Desjardins from DeepMind, Dr. John Smith from IBM Research, Dr. Orhan Firat from Google, and many other staff members, fellow students and visitors in the MILA lab.

Lastly, I would like to thank my parents who support me during my academic career all these years since I left China for Finland almost 8 years ago. They encourage me to explore my potential and pursue my dream. I would also like to thank my girlfriend Marie whom I met in Montréal and who has the courage to follow me to the strange land of San Francisco to start a new life with me.



# Chapter 1

---

## MACHINE LEARNING

Machine learning, a subject grown out of the intersection of statistics, psychology, neuroscience and artificial intelligence, has prospered during the past few decades. The first chapter looks at its historical development, reviews its core theory, exemplifies some of its most popular models, and exposes the computational challenge ahead.

### 1.1. MACHINE LEARNING AND ITS RELATION WITH OTHER SUBJECTS

Machine learning studies algorithms that learn patterns from data. Under the assumption that there exists an unknown data generating process, the purpose of learning is to uncover some aspects of it based on data samples it generates. Since the same data may be generated by any process, learning is only possible when some assumptions on the data generating process is made. This assumption is called the inductive bias that represents a hypothesis space where a learning algorithm searches to find the best possible hypothesis, or model, that explains the data. The need for establishing a hypothesis space is best demonstrated by the famous no-free-lunch theorem ([Wolpert, 1996](#)). The theorem shows that there is no universal best model. The reason is that one set of assumptions that works well in one domain may work poorly in another. This theorem diminishes the hope of finding a general-purpose learning algorithm that is able to solve all problems without any tuning. It motivates the necessity of injecting assumptions or prior knowledge of human experts into the effort of making an effective learning algorithm. In other words, learning requires prior knowledge.

Historically, machine learning grew out of the subject of artificial intelligence. In artificial intelligence, the goal is to construct an intelligent agent that possesses some human-like skills. To achieve that ambitious goal, an intelligent agent must be able to understand the surroundings and to take actions. This requires the skills of knowledge representation, reasoning, planning, and robotic control to be practically realizable. So far this goal remains elusive. Machine learning takes a step back and focuses on knowledge representation and reasoning. This was unified under the theory of probability by [Pearl \(1988\)](#). Knowledge is built into a probabilistic model and decision making is formalized by probabilistic inference.

Another line of development for machine learning grew out of the goal of understanding the working mechanism of the human brain in psychology and computational neuroscience. One of the earliest development in this direction is the McCulloch-Pitts Neuron (McCulloch and Pitts, 1943a) where a biological neuron is mathematically simplified in a spectacular fashion and still proved to be useful in solving some difficult tasks. This type of simple neuron, when organized in a hierarchy, has claimed many of the state-of-the-art results on machine learning benchmarks to date. Computational neuroscience is exemplified by biological inspired models such as Hierarchical Temporal Memory (HTM) (Hawkins and Blakeslee, 2004) and Hierarchical Model and X (HMAX) (Riesenhuber and Poggio, 1999). HTM consists of artificial neurons that are more biologically plausible and has been proved to be useful at modelling time series. HMAX has been widely accepted as the standard model of visual cortex. The recent study of Bengio et al. (2015) shed some light on the connection between synaptic weight updates and a particular form of variational EM algorithm.

## 1.2. FORMALIZING LEARNING

Following tradition, learning is divided into three categories. Supervised learning, unsupervised learning and reinforcement learning. Supervised and unsupervised learning are introduced below.

Supervised learning, such as classification and regression, focuses on the prediction of a target  $\vec{y}$  given an observation  $\vec{x}$  where  $(\vec{x}_i, \vec{y}_i) \sim P(\vec{x}, \vec{y})$ .  $P(\vec{x}, \vec{y})$  is an unknown data generating distribution. In regression,  $\vec{y}$  represents either a real-value scalar or a vector. In classification,  $\vec{y}$  usually represents a discrete code indicating which class  $\vec{x}$  belongs to. Predicting  $\vec{y}$  given  $\vec{x}$  is usually done by estimating  $P(\vec{y}|\vec{x})$  using training pairs  $(\vec{x}_i, \vec{y}_i)$ .

Unsupervised learning is more diversified. The majority of problems in this category can be considered as a problem of density estimation: given observations  $\vec{x} \sim P(\vec{x})$ , estimate implicitly or explicitly the density  $P(\vec{x})$ .

There are two views of learning: Bayesian and frequentist. The two views differ in the way the concept probability is treated. Bayesian learning insists on using probability as a way to measure the degree of belief. On the other hand, frequentist learning treats probability as the frequency of occurrence for certain events. This yields different treatment of parameters specifying underlying probabilistic models. In Bayesian learning, there exists a probability distribution on the parameters. As training data are seen, the degree of belief, or the probability distribution, on model parameters shall be updated. On the other hand, frequentist learning assumes an unknown single value of a parameter. As training data are seen, a point estimation of a parameter is obtained by performing optimization with respect to some criterion. The difference between these two learning paradigms is illustrated below in terms of density estimation.

### 1.2.1. Basic idea of Bayesian learning

Although not directly used in the scope of this thesis, the important idea of Bayesian learning is briefly outlined below.

Given a training set  $\mathcal{D}$ , the fully-fledged Bayesian models  $P(\vec{x})$  as a marginalization over all possible parameters  $\theta$ .

$$P(\vec{x}|\mathcal{D}) = \int_{\theta} P(\vec{x}|\theta)P(\theta|\mathcal{D})d\theta \quad (1.2.1)$$

where the posterior over the parameter  $\theta$  is defined as

$$P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)P(\theta)}{P(\mathcal{D})} \quad (1.2.2)$$

One has to specify the parameterizations of both the prior  $P(\theta)$  and the likelihood  $P(\mathcal{D}|\theta)$  in order to compute the posterior. When the marginalization in (1.2.1) is analytically intractable, one could fall back to the maximum a posteriori (MAP) estimation

$$\theta_{\text{MAP}}^* = \arg \max_{\theta} P(\mathcal{D}|\theta)P(\theta) \quad (1.2.3)$$

In MAP,  $P(\vec{x}|\mathcal{D})$  is approximated by

$$P(\vec{x}|\mathcal{D}) \approx P(\vec{x}|\theta_{\text{MAP}}^*) \quad (1.2.4)$$

(1.2.4) is a good approximation when the posterior over  $\theta$  is unimodal and  $\theta_{\text{MAP}}^*$  is the mode. As more data arrive, MAP in (1.2.4) and Bayesian prediction in (1.2.1) become closer, as MAP's competing hypotheses become less likely. As is usually the case, performing numerical optimization is relatively easier than estimating the integral.

### 1.2.2. Frequentist learning

Frequentist learning focuses on the point estimation of the model parameters. One of the most popular frameworks is maximum likelihood estimation (MLE).

$$\theta_{\text{MLE}}^* = \arg \max_{\theta} P(\mathcal{D}|\theta) \quad (1.2.5)$$

MLE can be justified from various perspectives. MLE is closely related to the well-known Kullback-Leibler (KL) divergence that measures the discrepancy between two probability distributions. The KL divergence between the unknown data generating distribution  $P(\vec{x})$  and the model  $P_{\theta}(\vec{x})$  is defined as

$$\text{KL}(P(\vec{x})||P_{\theta}(\vec{x})) = \int_{\vec{x}} P(\vec{x}) \log \frac{P(\vec{x})}{P_{\theta}(\vec{x})} d\vec{x} \quad (1.2.6)$$

$$= \int_{\vec{x}} P(\vec{x}) \log P(\vec{x}) - \int_{\vec{x}} P(\vec{x}) \log P_{\theta}(\vec{x}) \quad (1.2.7)$$

where the second term on the right is called cross-entropy between  $P(\vec{x})$  and  $P_\theta(\vec{x})$ . The first term does not depend on  $\theta$ . Using  $N$  training examples from  $\mathcal{D}$ , The empirical minimization of the KL is written as

$$\hat{\theta}_{\text{KL}}^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N \log \frac{P(\vec{x}_i)}{P_\theta(\vec{x}_i)} \quad (1.2.8)$$

where  $\vec{x}_i \sim P(\vec{x}_i)$ . Under the assumption that the examples in  $\mathcal{D}$  are independently identically distributed (i.i.d.), the empirical risk minimization of log-likelihood can be written as follows:

$$\begin{aligned} \hat{\theta}_{\text{MLE}}^* &= \arg \max_{\theta} \log \left( \prod_i^N P_\theta(\vec{x}_i) \right) \\ &= \arg \max_{\theta} \sum_i^N \log P_\theta(\vec{x}_i) \\ &= \arg \max_{\theta} \frac{1}{N} \sum_i \log \frac{P_\theta(\vec{x}_i)}{P(\vec{x}_i)} \\ &= \arg \min_{\theta} \frac{1}{N} \sum_i \log \frac{P(\vec{x}_i)}{P_\theta(\vec{x}_i)} \\ &= \hat{\theta}_{\text{KL}}^* \end{aligned} \quad (1.2.9)$$

Thus maximizing the log-likelihood under the i.i.d. assumption is equivalent to minimizing the KL divergence between the model distribution and the true unknown distribution.

MLE suffers from overfitting (detailed in section 1.3) when training data is scarce or the model  $P_\theta(\vec{x}_i)$  is too complex. For instance, consider a dataset that only contains a few coin flips that are all heads.  $\theta$  denotes the probability of a coin flip being heads. MLE results in  $\theta = 1$  since only heads are observed. This estimated value is highly unlikely and overconfident. With large datasets, this kind of overfitting is still possible when a model with a large number of parameters is used. Therefore in practice, it is common to use a regularized version of MLE

$$\hat{\theta}_{\text{MLE}_{reg}}^* = \arg \max_{\theta} [\log P(\mathcal{D}|\theta) + \lambda \phi(\theta)] \quad (1.2.10)$$

where a regularization term  $\phi(\theta)$  penalizes the model complexity and  $\lambda$  weighs the penalization. There exists a connection between  $\hat{\theta}_{\text{MLE}_{reg}}^*$  and  $\theta_{\text{MAP}}^*$  in (1.2.3).

$$\theta_{\text{MAP}}^* = \arg \max_{\theta} P(\mathcal{D}|\theta)P(\theta) = \arg \max_{\theta} [\log P(\mathcal{D}|\theta) + \log P(\theta)] \quad (1.2.11)$$

Therefore,  $\theta_{\text{MAP}}^*$  is equivalent to  $\hat{\theta}_{\text{MLE}_{reg}}^*$  and the regularization term in  $\hat{\theta}_{\text{MLE}_{reg}}^*$  is interpreted as the log of the prior  $P(\theta)$ .

As an estimator, the unbiasedness of MLE varies from case to case, for example, in estimating the mean  $\mu$  and the standard deviation  $\sigma$  in a univariate Gaussian distribution



$\mathcal{N}(\mu, \sigma)$ ,  $\hat{\mu}_{\text{MLE}} = \frac{1}{N} \sum_n x_i$  is unbiased but  $\hat{\sigma}_{\text{MLE}}^2 = \frac{1}{N} \sum_n (x_i - \hat{\mu}_{\text{MLE}})^2$  is biased. MLE has some desirable properties as a default estimator in learning. Asymptotically, MLE is consistent, meaning  $P_\theta(x) \rightarrow P(x)$  as  $N \rightarrow \infty$ . It is also asymptotically optimal or efficient in that it has the lowest variance among all the estimators. However MLE only provides a point estimation. Compared with Bayesian learning, it does not give any measure of uncertainty in the guess. Furthermore, MLE computation may also be expensive with complicated probabilistic models, which is usually the case in machine learning with high dimensional data. Other point estimators exist in machine learning. More tractable alternatives have been proposed such as Pseudo likelihood (Besag, 1975), ratio matching (Hyvärinen, 2007), generalized score matching (Lyu, 2009), and minimum probability flow (Sohl-Dickstein et al., 2009) and moment matching (Li et al., 2015).

### 1.2.3. Parametric and non-parametric learning

Orthogonal to the categorization of learning into supervised and unsupervised, an alternative view is to differentiate learning algorithms between parametric and non-parametric. In parametric learning, the parameter has a fixed size that is independent of number of training examples  $N$ . Making predictions does not involve any explicit use of the historical data any more since  $P(\vec{x}|\theta)$  is fully realizable by a function with the learned parameter  $\theta$ . One could argue that assuming any particular parameterization of  $P(\vec{x}|\theta)$  raises questions on the reliability of the assumption. This is indeed one shortcoming of parametric learning.

Conversely, in non-parametric learning,  $P(\vec{x})$  does not have a prefixed number of parameters and may grow automatically with more data. Indeed, most iconic non-parametric models such as K nearest neighbor classifiers and Parzen density estimators directly memorize the entire historical data. In particular, Parzen density estimator defines the probability density

$$p(\vec{x}) = \frac{1}{N} \sum_{n=1}^N \frac{1}{2\pi h^2} \exp\left(-\frac{\|\vec{x} - \vec{x}_n\|^2}{2h^2}\right) \quad (1.2.12)$$

at a particular point  $\vec{x}$  as a linear combination of Gaussians that are centered at each training example  $\vec{x}_i$ . Similarly, one can define a Parzen density estimator for a conditional density in (1.2.13) through the Bayes rule.

$$p(\vec{y}|\vec{x}) = \frac{p(\vec{x}, \vec{y})}{p(\vec{y})} = \sum_{n=1}^N w(\vec{x}_n) \frac{1}{2\pi h^2} \exp\left(-\frac{\|\vec{y} - \vec{y}_n\|^2}{2h^2}\right) \quad (1.2.13)$$

where

$$w(\vec{x}_n) = \frac{\exp\left(-\frac{\|\vec{x} - \vec{x}_n\|^2}{2h^2}\right)}{\sum_{n=1}^N \exp\left(-\frac{\|\vec{x} - \vec{x}_n\|^2}{2h^2}\right)} \quad (1.2.14)$$

The conditional kernel density in (1.2.13) can be considered as a mixture of Gaussians where each Gaussian is defined on  $\vec{y}_n$  and each has a mixing coefficient  $w(\vec{x}_n)$  defined on  $\vec{x}_n$ . As is the requirement of this mixture, the mixing coefficients should sum to 1, which is guaranteed by (1.2.14). Both (1.2.12) and (1.2.13) have been extensively studied in the statistics community. One can show that they converge to the true distribution as  $N \rightarrow \infty$  and  $h \rightarrow 0$  in the appropriate rate.

The non-parametric kernel density estimator suffers from the curse of dimensionality. This is further elaborated in section 2.2. Furthermore, making a prediction using all (or even a small yet significant subset) of the historical examples raises a serious computational issue when they are applied to problems involving a large number of examples. Both issues can be alleviated by using parametric models in the inner loop of non-parametric learning.

Modern neural networks are often trained in such a hybrid approach. Hyper-parameters such as its depth and its layer size are not fixed and could be tuned with cross-validation as data grow. Particular choice of hyper-parameters results in a fixed number of parameters such as its weights and biases which are trained in a parametric fashion.

### 1.3. LEARNING TO GENERALIZE

Generalization is central in learning. If learning successfully recovers the underlying data generating process, or it successfully captures the underlying factors, generalization is achieved. Then given new examples that are not seen in learning, the learner is able to make good decisions on it. Both supervised learning and unsupervised learning require the ability of generalization.

#### 1.3.1. Generalization in supervised learning

In supervised learning, a model is presented with examples that contain many variations. Some variations are prevalent, and others are occasional and may be caused by sampling noise. With limited learning capacity, for example limited number of parameters, a model should focus on capturing variations that are most relevant to the learning targets in order to generalize well. However, too limited capacity may lead to underfitting where the model is too weak to even predict targets well on the training examples. Increasing model capacity seems a natural choice in this situation. Yet as the capacity of a model increases, it not only learns to associate targets with the prevalent patterns but also with noise. Overfitting happens when a model fully captures all the variations in the training examples, yielding superb predictions on the training examples, yet later performing poorly on the unseen data. Thus, controlling model capacity is key to generalization.

### 1.3.2. Generalization in unsupervised learning

In unsupervised learning, overfitting and underfitting also apply. In the case of density estimation, underfitting happens when the density model is too simple to faithfully represent the data. For instance, one uses a single Gaussian to fit data that is generated by a mixture of Gaussians. On the other hand, overfitting happens when the probability mass is only assigned to the training data, resulting in very little probability mass being assigned to data unseen in the training.

### 1.3.3. Formalizing the generalization error

Learning can be considered as approximating a target function  $f^*$  with an estimated function  $\hat{f}$ . Then the generalization error could be formalized by a mean square error (MSE)  $\mathbb{E}[(\hat{f}(x) - f^*(x))^2]$  that measures the difference between the two on each data point  $x \sim P(x)$ . Consider getting multiple data samples  $\mathcal{D}$  from the unknown data generating distribution  $P(x)$ . An estimation  $\hat{f}$  of  $f^*$  is based on a particular data sample  $\mathcal{D}$ . Since  $\mathcal{D}$  is considered as a random variable, so is  $\hat{f}$ . So  $\bar{f} = \mathbb{E}[\hat{f}]$  is the expectation on the random variable  $\hat{f}$ . All the expectations are taken under  $P(x)$ . Thus the generalization error  $\mathbb{E}[(\hat{f} - f^*)^2]$  is decomposed as follows

$$\mathbb{E}[(\hat{f} - f^*)^2] = \mathbb{E}[(\hat{f} - \bar{f}) + (\bar{f} - f^*)]^2 \quad (1.3.1)$$

$$\begin{aligned} &= \mathbb{E}[(\hat{f} - \bar{f})^2] + 2(\bar{f} - f^*)\mathbb{E}[\hat{f} - \bar{f}] + (\bar{f} - f^*)^2 \\ &= \mathbb{E}[(\hat{f} - \bar{f})^2] + (\bar{f} - f^*)^2 \\ &= \text{var}[\hat{f}] + \text{bias}^2(\hat{f}) \end{aligned} \quad (1.3.2)$$

It may seem a good decision to pick an unbiased estimator whenever it is possible. However, the bias-variance decomposition in (1.3.2) indicates that it might be wise to use a biased estimator as long as the variance is sufficiently reduced, assuming the goal of minimizing the mean square error. The decomposition also indicates that a model that is too complex, having too many parameters, yields low bias but higher variance. On the other hand, a model that is too simple yields low variance but larger bias.

### 1.3.4. Model selection

The urge of seeking good generalization leads to the important issue of model selection. To estimate the generalization performance of a model, one typically divides the entire dataset into two parts,  $\mathcal{D}_{train}$  and  $\mathcal{D}_{test}$ . The training and the model selection are performed on  $\mathcal{D}_{train}$  while  $\mathcal{D}_{test}$  is solely used to evaluate the generalization performance of the selected model. Several methods can be used to select a good model.

#### 1.3.4.1. Bayesian model averaging

In theory, the choice of models may be considered as a random variable. This is indeed the case in Bayesian model averaging. Following the Bayesian learning in (1.2.1)

$$P(\vec{x}|\mathcal{D}) = \int_k P(\vec{x}|M_k, \mathcal{D})P(M_k|\mathcal{D})dM_k$$

The posterior over a particular choice of model is

$$P(M_k|\mathcal{D}) \propto P(\mathcal{D}|M_k)P(M_k)$$

and the model for the likelihood is

$$P(\mathcal{D}|M_k) = \int_{\theta_k} P(\mathcal{D}|\theta_k)P(\theta_k|M_k)d\theta_k$$

For the same reason of requiring to compute marginalizations, Bayesian model averaging is rarely practical.

#### 1.3.4.2. Cross-validation

A practically useful method in model selection is cross-validation. It is intuitively straightforward and easy to implement. It has been proved to work well in practice in numerous settings in machine learning. The standard procedure is to further divide  $\mathcal{D}_{train}$  into multiple subsets. Training is then performed on some of the subsets and validation is performed on the others. It is critical not to use validation subsets in training. The validation subsets are only used to perform model selection, for example. In cross-validation, one treats parameters and hyper-parameters in an equal footing in the sense that different configurations of both indicate different models. Performing model selection on the validation subsets amounts to picking a configuration of both of them that gives the best performance with respect to a chosen criterion, such as classification error, likelihood, or average squared error.

#### 1.3.4.3. Bagging and boosting

Bagging is short for bootstrap aggregation. In bootstrapping, one samples  $k$  batches from the given dataset with replacement and fits a different model on each of the  $k$  sets. In aggregation, the prediction is given by taking an average over all predictions from  $k$  models. By averaging, the variances in the predictions of individual models tend to cancel out.

Boosting is another good way to combine models. Boosting fits a series of weak models that, when combined together, are significantly better than individuals. The individual models are coordinated such that model  $k$  should focus on the part that previous  $k - 1$  models get wrong.

## 1.4. THE CURSE OF DIMENSIONALITY AND ITS IMPLICATION

Along with the concept of generalization, the curse of dimensionality is a related major difficulty faced by machine learning. In a low dimensional space, regions in the space are typically well covered by training examples. In that case, most machine learning algorithms with enough capacity, non-parametric or not, give reasonable answers as enough information of the data distribution is indeed available. As the dimensionality increases, the story of fitting a generalizable model becomes dramatically different. First of all, there will not be enough data to well cover the entire input space. Secondly, most test examples are very different from the training examples. In fact, the neighborhood of a test example is most likely covered by insufficient number of training examples, as argued in [Bengio et al. \(2010a\)](#). As a result, generalization becomes a problem of extrapolation rather than interpolation.



# Chapter 2

---

## REPRESENTATION LEARNING

A powerful learning algorithm must rely on a proper representation of data to solve problems. Representations could be made so powerful such that even a linear classifier such as a perceptron ([Rosenblatt, 1957](#)) is sufficient to solve an otherwise non-linearly separable task. However, transforming data from the input space to such a feature space is challenging in that one does not know what makes a good representation of the given task. Learning such representations automatically from the data is the central theme of machine learning and neural network research. The promise is machines can extract knowledge from data with little human intervention. This chapter follows Chapter 1, and delves into the subject area of representation learning with the focus on using neural networks that have been proved to be superior in solving perception related AI tasks. Important families of neural network models are reviewed in order to facilitate the understanding of further chapters.

### 2.1. CHALLENGING PROBLEMS IN THE AI-SET

Machine learning algorithms have been successfully applied to various domains. Yet there is one particular subset of problems that remains challenging. This subset of problems require human-like intelligent behavior, such as visual perception, auditory perception, planning and control. As argued by [Bengio and LeCun \(2007\)](#), this type of problem, dubbed the AI-set, typically involves handling perceptual signal that contains a great amount of variations. Let us consider, for instance, the task of object recognition where the same type of object may appear to be vastly different depending on physical factors such as the lighting condition, the orientation and angle of the camera, the texture, the shape, the scale, background occlusion. The number of such combined variations could easily be exponential. Therefore the solution to such problems inevitably requires learning algorithms to be able to capture a highly varying function under the constraints:

- (1) They do not need exponential number of training examples to generalize well on a problem with exponentially many variations.
- (2) The learning should require as little human involvement as possible.

- (3) The computational time performed at training and recognition should be able to scale sub-linearly, or at most linearly with the size of the problem.

As one example from the AI-set, let us consider the problem of object recognition in static images. This problem is difficult because of the factors of variations it embraces. The pixels of a typical photograph will depend on the scene type and geometry, the number, shape and appearance of objects present in the scene, their 3D positions and orientations, as well as effects such as occlusion, shading and shadows. All those factors interact with each other in an intricate way, making the true identity of an object a highly nonlinear function of the input pixels.

In order to explicitly model a potentially exponential number of variations in the data, one needs exponential number of examples that capture these variations. However, this is hardly practical as one rarely has a collection of training cases whose number comes even close to exponential. There is another more severe problem. Learning becomes even more impractical if the complexity of the model needs to grow exponentially to accommodate the growth in variations of the data. Many popular machine learning models fare poorly in that they require exponentially many parameters to generalize.

Let us consider two examples. Kernel machines with local kernels, such as SVMs, need as many kernels as the number of variations to well cover the input space because most kernels are local and shortsighted (Bengio et al., 2006). Tree-based models divide input space into cells, also requiring as many cells as the number of variations to generalize well (Bengio et al., 2010b). Both of them, in a way, suffer severely from the curse of dimensionality.

## 2.2. LEARNING HIERARCHICAL AND DISTRIBUTED REPRESENTATIONS

To fight the exponential growth of the number of variations, a distributed representation becomes invaluable. The notion of distributed representation was first introduced by Hinton (1986). The underlying assumption behind distributed representations is that the observed data were generated by the interactions of many unknown factors, and that what is learned about a particular factor from some configurations can often generalize to other, unseen configurations of the factors. In contrast to SVMs and trees whose representations are formed by looking in a locally defined neighbourhood, neural networks still learn representations from local examples but local examples have a knock-on effect in the learned representations that are far away from this neighbourhood, even in the part of the input space where no training examples are observed. This surprising generalization property relies on the ability to discover the underlying factors from the data. If the true factors were learned, training examples merely consist of some particular configurations of those factors, while generalization to unseen examples is achieved if factors are configured and composed differently according to the trained model.



Distributed representations clearly have some advantages in tackling problems in the AI-set. The statistical efficiency can be further improved if it is organized in a hierarchy. Deep representation learning (Bengio et al., 2013) goes in this direction. It adds the assumption that factors are organized into different layers, corresponding to different level of abstractions and compositions. Higher level representations are obtained by transforming lower level representations. This is a plausible assumption in solving problems in the AI-set. Let us again take the aforementioned example of object recognition. The underlying factors affecting object recognition can be naturally organized into a hierarchy. On the lowest level, factors such as edges and corners are learned from raw image pixels. The second level composes simple geometric shapes from the first level. The third level uses simple shapes to form object silhouettes. As more and more levels are added, the representations become more and more specialized and attached to a particular type of object, making efficient object recognition feasible. Another example of problems in the AI-set is understanding the meaning of a sentence. A sentence is naturally decomposed into elements such as letters, words, phrases whose complexity grows as they become more and more abstract. A deep and distributed representation is able to express such a sentence in a compact and efficient way by sharing and reusing basic elements in the layer below.

Neural networks and its variants, including many types of graphical models, possess desirable properties in terms of the three requirements to be considered as viable candidates to tackle the problems in the AI-set. More specifically, they enjoy distributed representations that could be easily organized in a hierarchical fashion. They could easily incorporate generic AI priors to automatically discover the underlying factors of variations in the dataset while requiring very little human intervention. Although the computational time scales differently from case to case, in general, they enjoy efficient computation and a good solution can be found relatively efficiently by stochastic optimization. The computational matter shall be discussed in details shortly.

This following sections offer an introduction to several types of neural network models that are capable of learning distributed and hierarchical representations. Three of them – feedforward neural networks, recurrent neural networks, and auto-encoders are closely related to the rest of the thesis. Although arguably less active in research, probabilistic graphical models made by neural networks are also included. The focus is on the formulation of models. The issue of computation is deferred to section 2.7. Historically, the term “neural networks” and the term “multilayer perceptrons” (MLPs) were used interchangeably.

## 2.3. FEEDFORWARD NEURAL NETWORKS

Feed-forward neural networks (Widrow and Lehr, 1990) is one of the earliest inventions that are able to learn a hierarchical and distributed representation of inputs. It is shown in Figure 2.1. It has a strictly feed-forward organization in that information flows only in one

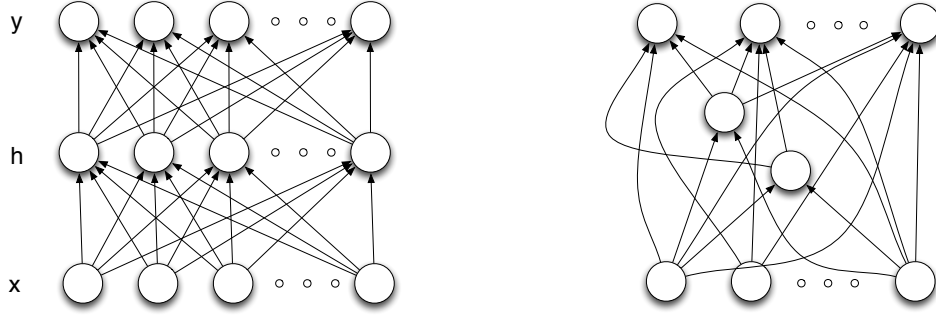


FIG. 2.1. *Left:* A feed-forward neural network with one input layer, one hidden layer, and one output layer. *Right:* A feed-forward neural network in a general form whose organization is not according to layers.

fixed direction and there is no loop. Traditionally, the structure of the MLP is organized into one input layer, one or many hidden layers and one output layer. MLPs build upon linear or generalized linear models and expand the model capacity by learning a collection of basis functions  $\phi_j(\vec{x})$ . Denote  $\vec{y}_i$  one of the outputs from the last layer of a three-layer MLP, and  $\theta = \{\vec{w}^{(1)}, \vec{w}^{(2)}\}$  the parameters (including both weights and biases) of hidden layer and output layer, the last layer of an MLP implements the following function

$$\vec{y}_i(\vec{h}; \vec{w}^{(2)}) = f\left(\sum_j \vec{w}_j^{(2)} \phi_j(\vec{h})\right) \quad (2.3.1)$$

where  $f$  represents an arbitrary fixed function such as an identity function  $x = I(x)$ , or sigmoid( $x$ ) =  $\frac{\exp(x)}{1+\exp(x)}$  commonly used in binary classifications, or softmax( $\vec{x}_i$ ) =  $\frac{\exp(\vec{x}_i)}{\sum_j \exp(\vec{x}_j)}$  commonly used in multi-class classifications. Similarly, an output from a hidden unit  $j$  is defined as

$$\phi_j(\vec{x}; \vec{w}^{(1)}) = f\left(\sum_k \vec{w}_k^{(1)} \vec{x}_k\right) \quad (2.3.2)$$

Such a two-layer model is easily generalized to the one with multiple layers. In MLPs, each hidden unit has a fixed capacity coming on the weights associated with it. A hidden unit only captures one aspect of the data. This aspect could be either a higher order correlation that involves all the inputs a unit sees and it could also be a lower order correlation that involves a small subsets of it, depending on how its weights are being set in learning. Once each hidden unit has been trained, it responds to a subset of configurations of inputs. Some inputs yield high value output while some yield lower value. By cascading such hidden units, each with a limited degree of freedom, MLPs are able to capture possibly very high order dependencies.

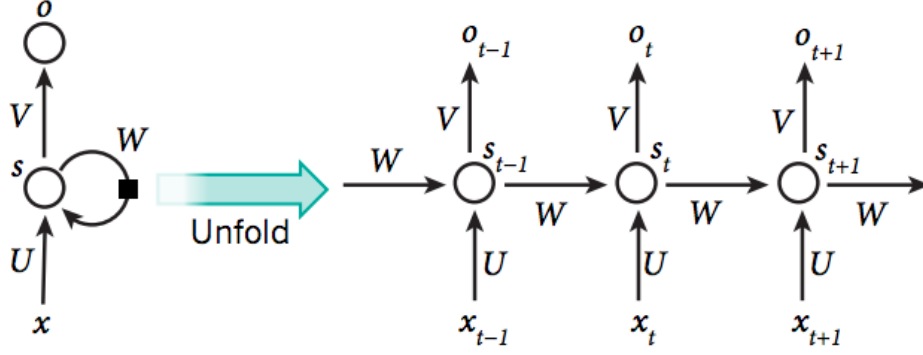


FIG. 2.2. *Left:* An RNN with one recurrent layer, one input layer and one output layer. *Right:* An RNN unfolds itself over time steps to form a feed-forward model with shared hidden layers. The figure is credited by *LeCun et al. (2015)*

## 2.4. RECURRENT NEURAL NETWORKS

Recurrent neural networks (RNNs) have regained their popularity in recent years as the standard machine learning model for time series such as videos, speech and texts. In Figure 2.2, inputs  $\mathbf{x}$  is composed by a time series  $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T$  with  $T$  time steps in total. In each time step, the following parameterization is being applied on  $\mathbf{x}_t$

$$\mathbf{s}_t = f(U\mathbf{x}_t + W\mathbf{s}_{t-1}) \quad (2.4.1)$$

$$\mathbf{o}_t = g(V\mathbf{s}_t) \quad (2.4.2)$$

with bias folded into the weight matrices of  $U$ ,  $W$  and  $V$ , and where  $f$  and  $g$  denote element-wise nonlinearities. It is evident from Figure 2.2 that the parameters are shared over time. This makes RNNs particularly suitable to perform computation on inputs of arbitrary length  $T$  without altering the number of parameters. Once unfolded, it is treated as a feed-forward neural network. Similar to the universal approximator property of feedforward neural networks from *Stinchcombe and White (1989)*, *Siegelmann and Sontag (1995)* shows that, for any computable function, there exists an RNN with a finite number of recurrently connected units that is able to compute it. Such theory, however, only asserts the existence rather than their trainability and therefore has little relevance in practice.

RNNs are notoriously difficult to train in practice due to the numerical attenuation of signal over long period of time (*Bengio et al., 1994*). To illustrate this with three time steps, compute the gradient of  $U$  with backpropagation

$$\frac{\partial o_{t+1}}{\partial U} = \frac{\partial o_{t+1}}{\partial s_{t+1}} \frac{\partial s_{t+1}}{\partial U} \quad (2.4.3)$$

$$+ \frac{\partial o_{t+1}}{\partial s_{t+1}} \frac{\partial s_{t+1}}{\partial s_t} \frac{\partial s_t}{\partial U} \quad (2.4.4)$$

$$+ \frac{\partial o_{t+1}}{\partial s_{t+1}} \frac{\partial s_{t+1}}{\partial s_t} \frac{\partial s_t}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial U} \quad (2.4.5)$$

where the three terms capture respectively the short, intermediate and long-term dependencies. Each term is essentially performing a series of matrix-vector multiplication, which could in principle lead to exploding or vanishing gradients, depending on the singular values of those matrices.

The work in this thesis uses extensively Long-short term memory (LSTM) recurrent neural networks, first introduced in [Hochreiter and Schmidhuber \(1997a\)](#), and Gated Recurrent Units (GRUs) from [Cho et al. \(2014\)](#). Compared with the formulation in Figure 2.2, they use gates to create shortcuts among distant time steps to carefully modify and preserve the long-term memory. Chapter 5 contains detailed explanation of LSTMs and Chapter 9 for GRUs.

## 2.5. AUTO-ENCODERS

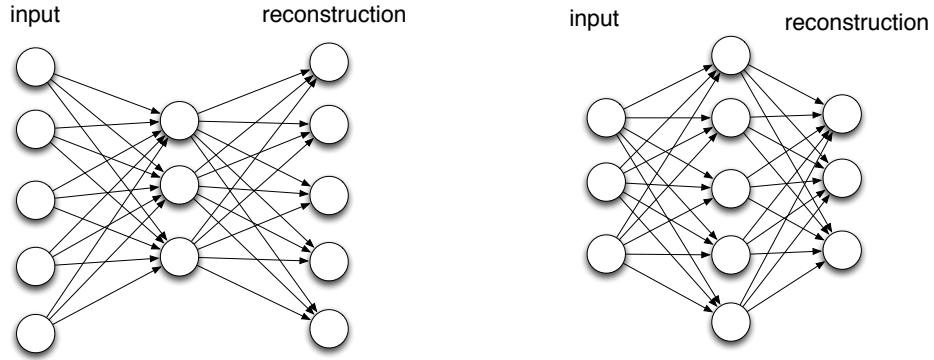


FIG. 2.3. *Left:* A standard auto-encoder must have an information bottleneck in the hidden layer to avoid learning the trivial identity reconstruction function. *Right:* A regularized auto-encoder that can have an over-complete hidden layer, such as denoising auto-encoders, and contractive auto-encoders since it cannot learn the identity function.

Unlike supervised MLPs, auto-encoders are capable of learning a hierarchical and distributed representation of inputs by performing reconstruction. It is thus unsupervised. Figure 2.3 shows two classes of such models. Given  $\vec{x} \in \mathbb{R}^d$ , a simple auto-encoder implements the following function

$$\vec{y} = f(W_1 \vec{x} + b_h) \quad (2.5.1)$$

and

$$\vec{z} = f(W_2 \vec{y} + b_o) \quad (2.5.2)$$

where  $\vec{x}$  denotes inputs,  $\vec{y}$  denotes activations from hidden units, and  $\vec{z}$  denotes final outputs, model parameters  $W_1, W_2, b_h, b_o$  are weights and biases on hidden and output units. Auto-encoders minimize a reconstruction cost that typically takes one of the following forms

$$\mathcal{L}_{mse} = \mathbb{E}_x[\|\vec{x} - \vec{z}\|^2] \quad (2.5.3)$$

or

$$\mathcal{L}_{ce} = \mathbb{E}_x[-\sum_i (\vec{x}_i \log(\vec{z}_i) + (1 - \vec{x}_i) \log(1 - \vec{z}_i))] \quad (2.5.4)$$

The cost  $\mathcal{L}_{mse}$  has a probabilistic interpretation that it is equivalent to the maximum likelihood estimation when assuming  $\vec{x}_i$  is a univariate Gaussian distribution given  $\vec{z}$ . Likewise, the cost  $\mathcal{L}_{ce}$  is based on the cross-entropy between two Binomial distributions and it requires both  $\vec{x}$  and  $\vec{z}$  taking values between 0 and 1 for it to make sense.

On the left of Figure 2.3 is a standard auto-encoder that has a hidden layer whose number of units is smaller than the number of inputs. On the right is a regularized auto-encoder that could afford to have an over-complete hidden layer. This distinction is made because a standard auto-encoder has the potential to learn an identity mapping from inputs  $\vec{x}$  to itself, while in a regularized one it is generally impossible since restrictions are added to the representations learned in the hidden layer. A typical example from the standard auto-encoder family is to set  $W_2 = W_1^T$  and to use linear activation on both hidden and output layer. One can show that this type of auto-encoder is capable of learning a span that is in the same linear space as principle components of  $\vec{x}$  (Baldi and Hornik, Baldi and Hornik).

The Denoising auto-encoder (DAE) (Vincent et al., 2008a) is a member of the regularized auto-encoder family. It differs from the standard auto-encoder in that a corrupted version of inputs  $\tilde{x}$  is used in  $\vec{y} = f(W_1\tilde{x} + b_h)$ . Intuitively, DAEs learn representations that are robust to the partial destruction of the inputs. Alternative regularizations are possible. For instance, in contractive auto-encoders (CAE), a regularization term  $\|J_y\|_F = \sum_{i,j} (\frac{\partial \vec{y}_i}{\partial \vec{x}_j})^2$  is added into the cost function to penalize the sensitivity of changes in hidden activations  $\vec{y}$  with respect to inputs  $\vec{x}$ . As the result of the competition between minimizing the reconstructions and penalizing the changes in the hidden units, CAEs and DAEs are able to learn the directions on the data manifold where changes are most significant.

## 2.6. GRAPHICAL MODELS BASED NEURAL NETWORKS

Probabilistic reasoning plays a central role in knowledge understanding and reasoning. Graphical models offer many tools to compactly represent a probabilistic relationship in the data and perform computation on it. There are two main families, directed graphical models (DGM), historically called Bayesian networks and undirected graphical models, historically called Markov random fields (MRF).

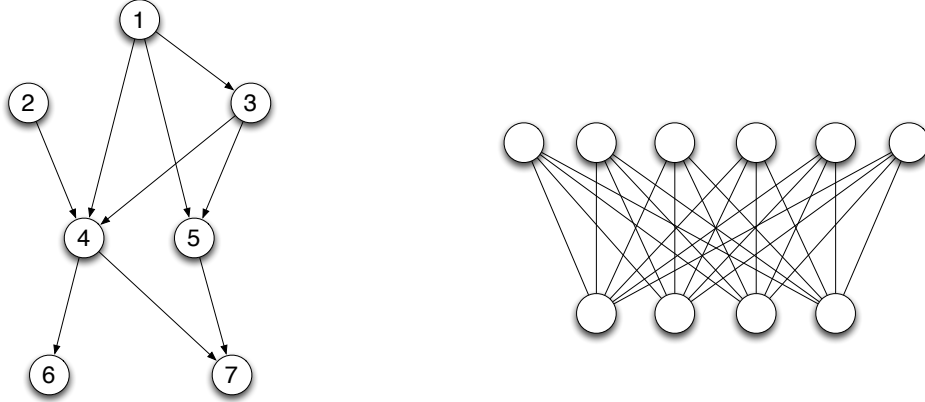


FIG. 2.4. *Left:* An example of directed graphical models. Edges are directed. The number in the nodes indicates a topological order. *Right:* The Restricted Boltzmann machine as an example of undirected graphical models. Edges are undirected.

### 2.6.1. Directed graphical models

A probability distribution can be expressed by a DGM in the form of

$$P(x_1, \dots, x_d) = P(x_1) \prod_{i=2}^d P(x_i | \text{par}(x_i))$$

where  $\text{par}(x_i)$  denotes all the parent nodes of  $x_i$ . It can be shown that such a probabilistic expression can be expressed equivalently in a form of a directed acyclic graph. An example is shown in Figure 2.4. The directed graph corresponds to  $P(x_1, \dots, x_7) = P(x_1)P(x_2)P(x_3|x_1)P(x_4|x_1, x_2, x_3)P(x_5|x_1, x_3)P(x_6|x_4)P(x_7|x_4, x_5)$  for the reason that they represent exactly the same set of conditional independencies between 7 variables.

The parametrization of individual factors  $P(x_i | \text{par}(x_i))$  is flexible. When nodes have discrete values, it can be parameterized by a probability table. In the continuous case, any continuous probability distribution can be adopted such as multivariate Gaussian distributions. It is even possible, sometimes more advantageous, to use neural networks to parameterize the factors, which offers great generalization ability over the traditional parameterizations.

To learn the parameters in the joint probability, maximum likelihood learning discussed in section 1.2 can be applied. When every node is observed, maximum likelihood learning over all the factors is reduced to the learning of local factors as the log probability of the joint distribution can be written in the following form

$$\log(P_{\theta}(\vec{x})) = \log(P_{\theta_1}(x_1) \prod_{i=2}^d P_{\theta_i}(x_i | \text{par}(x_i))) = \log(P_{\theta_1}(x_1)) + \sum_{i=2}^d \log(P_{\theta_i}(x_i | \text{par}(x_i)))$$

where each log factor in the sum can be maximized individually.

In theory, the capacity of a fully observable model can be made arbitrarily large by just adding edges between nodes. One faces the problem of explosion of parameters if the number of nodes goes large. One compact way to avoid the explosion of parameters while keeping the model expressive is to add latent variables that are not directly instantiated by the inputs. For instance, one could imagine a generation process where the identity of a hand-written digit is generated first, followed by generating the appearance of it given the identity. The latent variable could be used to capture such identity information as well as other high level concepts such as writing styles and habits, and furthermore such latent factors should be learned automatically from the data.

This is a popular choice in representation learning. For instance, commonly used principal components analysis (PCA), independent components analysis (ICA) and factor analysis (FA) can all be interoperated as directed graphical models with Gaussian latent variables. Sigmoid belief networks (Saul et al., 1996) goes further by stacking layers of binary latent variables and it can be shown that it is a universal approximator (Sutskever and Hinton, 2008). The major challenge of using latent variables is the difficulty of inference. This is due to the effect of explaining away (Pearl, 1988). It is explaining away that makes the directed models powerful in that it introduces lateral interactions between parents. It is also because of explaining away that exact inference is intractable except in a few very simple cases such as PCA and standard hidden Markov models (HMM).

Expectation maximization (EM) is a general framework to treat maximum likelihood learning in DGMs with latent variables. It is best seen from the decomposition of the likelihood into two terms in (2.6.1).

$$\log p(\vec{x}) = \mathcal{L}(q) + \text{KL}(q \parallel p) \quad (2.6.1)$$

$$\mathcal{L}(q) = \int q(\vec{z}) \log \frac{p(\vec{x}, \vec{z})}{q(\vec{z})} d\vec{z} \quad (2.6.2)$$

$$\text{KL}(q \parallel p) = - \int q(\vec{z}) \log \frac{p(\vec{z}|\vec{x})}{q(\vec{z})} d\vec{z} \quad (2.6.3)$$

EM consists of two steps. In the first step,  $\mathcal{L}(q)$  is maximized with respect to  $q(\vec{z})$  while keeping  $p(\vec{x}, \vec{z})$  fixed. This amounts to minimizing  $\text{KL}(q \parallel p)$ , making  $q(\vec{z})$  as close as possible to  $p(\vec{z}|\vec{x})$ . In the second step,  $\mathcal{L}(q)$  again is maximized but with respect to  $p(\vec{x}, \vec{z})$  while  $q(\vec{z})$  is kept fixed. If the E step makes  $\text{KL}(q \parallel p) = 0$ , the side effect of the M step will necessarily makes  $\text{KL}(q \parallel p) > 0$ . As a result, alternating the E and the M steps guarantees to increase  $\log p(\vec{x})$  till it converges. However, when the E step is not exact, this guarantee will be lost. Instead, only a lower bound is increased (Neal and Hinton, 1998).

### 2.6.2. Variational autoencoders

The major challenge of training the directed graphical model lies in the intractability of computing the true posterior distribution of latent variables given the observation  $p(\vec{z}|\vec{x})$ . In EM, both Eq. (2.6.2) and (2.6.3) are valid for any distribution  $q$ . Variational Autoencoders (VAEs) from [Kingma and Welling \(2013\)](#) proposed to use  $q(\vec{z}|\vec{x})$ . Following Eq. (2.6.1),

$$\log p(\vec{x}) = \mathcal{L}(q(\vec{z}|\vec{x})) + \text{KL}(q(\vec{z}|\vec{x}) \parallel p(\vec{z}|\vec{x})) \quad (2.6.4)$$

$$= E_{z \sim q}[\log p(\vec{x}|\vec{z}) + \log p(\vec{z}) - \log q(\vec{z}|\vec{x})] + \text{KL}(q(\vec{z}|\vec{x}) \parallel p(\vec{z}|\vec{x})) \quad (2.6.5)$$

$$= E_{z \sim q}[\log p(\vec{x}|\vec{z})] - E_{z \sim q}[\log q(\vec{z}|\vec{x}) - \log p(\vec{z})] + \text{KL}(q(\vec{z}|\vec{x}) \parallel p(\vec{z}|\vec{x})) \quad (2.6.6)$$

$$\geq E_{z \sim q}[\log p(\vec{x}|\vec{z})] - E_{z \sim q}[\log q(\vec{z}|\vec{x}) - \log p(\vec{z})] \quad (2.6.7)$$

$$= E_{z \sim q}[\log p(\vec{x}|\vec{z})] - \text{KL}(q(\vec{z}|\vec{x}) \parallel p(\vec{z})) \quad (2.6.8)$$

where  $\log p(\vec{x})$  is lower bounded only by the sum of  $E_{z \sim q}[\log p(\vec{x}|\vec{z})]$  and  $\text{KL}(q(\vec{z}|\vec{x}) \parallel p(\vec{z}))$  due to the drop of the non-negative term  $\text{KL}(q(\vec{z}|\vec{x}) \parallel p(\vec{z}|\vec{x}))$ . As one may realize from the above derivation that the VAE objective in Eq. (2.6.8) has very little to do with autoencoders in Section 2.5, and it just happens that the resulting lower bound contains the term  $q(\vec{z}|\vec{x})$  that may be parameterized by a neural network encoder, and the term  $p(\vec{x}|\vec{z})$  that may be parameterized by a neural network decoder. The use of “variational” in its name has its origin in the machine learning literature where Eq. (2.6.8) is referred as the variational lower bound of the MLE estimator. In the original work of [Kingma and Welling \(2013\)](#), the second term of the objective is made analytically tractable by assuming that the encoder function  $q(\vec{z}|\vec{x})$  is Gaussian with a diagonal covariance matrix to simplify the computation.

In theory, one could train VAEs with the above objective by using a stochastic approximation of the expectation term. This is very inefficient. Notice that the first term in Eq. (2.6.8) can be reformulated as

$$E_{z \sim q}[\log p(\vec{x}|\vec{z})] = E_{z' \sim \mathcal{N}(0,1)}[\log p(\vec{x}|f(z'))] \quad (2.6.9)$$

where  $\vec{z} = f(z')$  and  $f$  is a deterministic function of  $z'$  that is sampled from a normal distribution. This is referred as the “reparameterization trick” in [Kingma and Welling \(2013\)](#). It enables the backpropagation to run from the decoder all the way to the encoder without encountering any sampling path that blocks the gradient flow.

### 2.6.3. Markov random fields

MRFs are responsible for the resurgence of neural network research in 2006 with models such as Restricted Boltzmann Machines (RBMs) from [Hinton and Salakhutdinov \(2006b\)](#), Deep Belief Networks (DBNs) from [Hinton et al. \(2006a\)](#) and Deep Boltzmann Machines



(DBMs) from Salakhutdinov and Hinton (2009a). MRFs hold great promise in unsupervised learning. Over the years, they have become less popular as training such models is challenging where the gradient is often intractable and requires approximations that are by themselves computationally expensive. Unlike a DGM, an MRF represents a joint probability distribution using a product of potentials.

$$p(\vec{x}) = \frac{1}{Z} \prod_c \psi_c(\vec{x}_c) \quad (2.6.10)$$

where a potential  $\psi_c(\vec{x}_c)$  is defined on a group of variables within one of the maximal cliques  $c$ .  $Z$  is the normalization constant that ensures  $p(\vec{x})$  is well-defined. Contrary to the factor used in DGM, which is strictly a valid probability, a potential is not upper bounded as long as it is positive to ensure  $p(\vec{x}) > 0$ .

The maximum likelihood learning of an MRF is more complicated than that of a DGM because one has to explicitly deal with the normalization term. This is illustrated via an example below. Consider the restricted Boltzmann machines (RBM) in Figure 2.4. The joint probability is defined as

$$p(v, h) = \frac{\exp(-E_\theta(v, h))}{Z} \quad (2.6.11)$$

where  $Z = \sum_{v, h} \exp(-E_\theta(v, h))$  is the partition function, which ensures that  $p(v, h)$  is normalized. We can recover the marginal distribution  $p(v)$  by summing over the latent variables:

$$p(v) = \frac{1}{Z} \sum_h \exp(-E_\theta(v, h)) \quad (2.6.12)$$

It is also useful to define the Free-Energy function  $F_\theta(v) = -\log \sum_h \exp(-E_\theta(v, h))$ , which allows us to write  $p(v) = \frac{\exp(-F_\theta(v))}{Z}$

An RBM is further parameterized by the following energy function:

$$E_\theta(v, h) = - \sum_{i=1}^{n_v} \sum_{j=1}^{n_h} W_{ij} h_j v_i - \sum_j c_j h_j - \sum_i b_i v_i \quad (2.6.13)$$

which groups  $n_v$  visible units and  $n_h$  hidden units into two separate layers, interacting through the weight matrix  $W \in \mathbb{R}^{n_h \times n_v}$ . Connections between units of the same layer are prohibited, differentiating it from other more general Boltzmann Machines.  $b \in \mathbb{R}^{n_v}$  and  $c \in \mathbb{R}^{n_h}$  are the offsets of the visible and hidden units, and serve much the same purpose as MLPs. While the random variable  $v$  and  $h$  can belong to many probability distributions, we will focus here on the binary-binary RBM, where  $v \in \{0, 1\}^{n_v}$  and  $h \in \{0, 1\}^{n_h}$ .

$$p(h|v) = \frac{p(v, h)}{p(v)} = \frac{p(v, h)}{\sum_h p(v, h)} = \frac{\exp[-E_\theta(v, h)]}{\sum_h \exp[-E_\theta(v, h)]}$$

$$\begin{aligned}
&= \frac{\exp \left[ -\sum_j h_j (c_j + \sum_i W_{ij} v_i) \right] \exp \left( -\sum_i b_i v_i \right)}{\sum_h \exp \left[ -\sum_j h_j (c_j + \sum_i W_{ij} v_i) \right] \exp \left( -\sum_i b_i v_i \right)} \\
&= \frac{\prod_j \exp \left[ -h_j (c_j + \sum_i W_{ij} v_i) \right]}{\sum_h \prod_j \exp \left[ -h_j (c_j + \sum_i W_{ij} v_i) \right]} \\
&= \frac{\prod_j \exp \left[ -h_j (c_j + \sum_i W_{ij} v_i) \right]}{\prod_j \sum_{h_j} \exp \left[ -h_j (c_j + \sum_i W_{ij} v_i) \right]} \\
&= \prod_j \frac{\exp \left[ -h_j (c_j + \sum_i W_{ij} v_i) \right]}{1 + \exp \left[ -c_j - \sum_i W_{ij} v_i \right]}
\end{aligned}$$

$$p(h|v) = \prod_j p(h_j|v) \quad (2.6.14)$$

$$p(h_j = 1|v) = \text{sigmoid} \left( c_j + \sum_i W_{ij} v_i \right) \quad (2.6.15)$$

By symmetry of the energy function, we can write:

$$p(v|h) = \prod_i p(v_i|h) \quad (2.6.16)$$

$$p(v_i = 1|h) = \text{sigmoid} \left( b_i + \sum_j W_{ij} h_j \right) \quad (2.6.17)$$

We can derive the free-energy of the binary-binary RBM as follows:

$$\begin{aligned}
F_\theta(v) &= -\log \sum_h \exp^{-E_\theta(v,h)} \\
&= -\log \left[ \exp \left( -\sum_i b_i v_i \right) \sum_h \prod_j \exp \left[ -h_j \left( c_j + \sum_i W_{ij} v_i \right) \right] \right] \\
&= -\sum_i b_i v_i - \log \prod_j 1 + \exp \left[ -c_j - \sum_i W_{ij} v_i \right] \\
&= -\sum_i b_i v_i - \sum_j \log \left[ 1 + \exp \left( -c_j - \sum_i W_{ij} v_i \right) \right] \quad (2.6.18)
\end{aligned}$$

$$\begin{aligned}
-\log p(x; \theta) &= F(x) + \log Z \\
&= F(x) + \log \sum_x \exp [-F(x)] \\
-\frac{\partial \log p(x)}{\partial \theta} &= \frac{\partial F(x)}{\partial \theta} + \frac{1}{Z} \frac{\partial Z}{\partial \theta} \\
&= \frac{\partial F(x)}{\partial \theta} + \frac{1}{Z} \sum_x \frac{\partial \exp [-F(x)]}{\partial \theta}
\end{aligned}$$

$$\begin{aligned}
&= \frac{\partial F(x)}{\partial \theta} - \frac{1}{Z} \sum_x \exp[-F(x)] \frac{\partial F(x)}{\partial \theta} \\
&= \frac{\partial F(x)}{\partial \theta} - p(x) \sum_x \frac{\partial F(x)}{\partial \theta} \\
&= \frac{\partial F(x)}{\partial \theta} - \mathbb{E}_p \left[ \frac{\partial F(x)}{\partial \theta} \right]
\end{aligned} \tag{2.6.19}$$

The maximum likelihood update equations for  $\theta$ , at the  $t$ -th parameter update, becomes:

$$\theta_{t+1} \leftarrow \theta_t - \alpha \left( \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \left[ \frac{\partial F(x)}{\partial \theta} \right] - \mathbb{E}_p \left[ \frac{\partial F(x')}{\partial \theta} \right] \right) \tag{2.6.20}$$

---

**Algorithm 1** Stochastic Maximum Likelihood Update

---

Obtain (mini)-batch of training examples  $\mathcal{X}_t^+ = \{x \in \mathcal{D}\}$ .

Initialize (mini)-batch of samples  $x_t^{(0)}$  from  $\mathcal{X}_{t-1}$ , approx. samples of  $p_{t-1}$ .

**for**  $k = 0 : K$  **do**

    Generate (mini)-batch of samples  $h_t^{(k)} \sim p_t(h|v = x_t^{(k)})$ .

    Generate (mini)-batch of samples  $x_t^{(k+1)} \sim p_t(v|h = h_t^{(k)})$ .

**end for**

Define  $\mathcal{X}_t$  as the (mini)-batch of samples  $x_t^{(k)}$ .

$\theta_{t+1} \leftarrow \theta_t - \alpha \left( \frac{1}{|\mathcal{X}_t^+|} \sum_{x \in \mathcal{X}_t^+} \left[ \frac{\partial F(x)}{\partial \theta} \right] - \frac{1}{|\mathcal{X}_t|} \sum_{x' \in \mathcal{X}_t} \left[ \frac{\partial F(x')}{\partial \theta} \right] \right)$ .

---

There are many ways to perform the update rule in (2.6.20) by approximating the intractable expectation. One popular choice is stochastic maximum likelihood (Tieleman, 2008) detailed in Algorithm 1.

## 2.7. COMPUTATIONAL TECHNIQUES

Much of the research effort in Machine learning has been devoted to modelling. Equally important issues arise in computation. This section outlines important computational techniques for training modern neural networks.

### 2.7.1. Optimization

The first computational challenge is in optimization. Techniques in optimization are further divided into several types, depending on the type of models. SVMs were popular partly because they only require quadratic programming which is essentially a convex, usually constrained optimization problem. On the other hand, mainstream neural networks require solving highly non-convex problems, usually unconstrained. If the objective function is

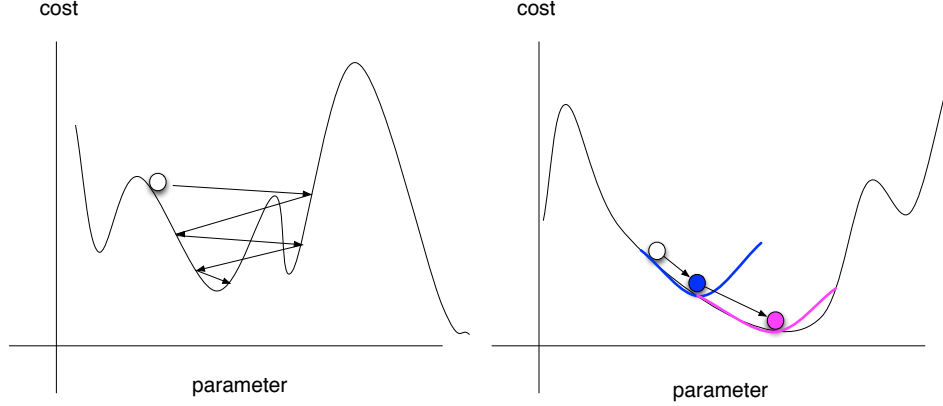


FIG. 2.5. *Left:* SGD uses noisy gradient to reach global minima. *Right:* Second order optimization methods use a local quadratic approximation and move to the minimum of a quadratic curve at each step.

smooth and differentiable, as is usually the case in neural networks, the best way is to take advantage of the gradient. To see this, use the definition of gradient when  $\Delta\theta \rightarrow 0$

$$f'(\theta)\Delta\theta = f(\theta + \Delta\theta) - f(\theta) \quad (2.7.1)$$

$f(\theta + \Delta\theta) > f(\theta) \rightarrow f'(\theta)$  and  $\Delta\theta$  are in the same direction, justifying the use of gradient descent. Therefore, in gradient descent, one uses  $f'(\theta)$  as the moving direction and the length of the move is decided by a line search. However, in practice, a technique called stochastic gradient descent (SGD) (Bottou, 2013) proves to be more efficient. SGD uses the following rule to update parameters in order to minimize  $f$ :

$$\theta_{t+1} = \theta_t - \alpha f'(\theta_t) \quad (2.7.2)$$

It is shown in Robbins and Monro (1951) that the convergence to a local minimum is guaranteed if  $\sum_{t=1}^{\infty} \alpha_t = \infty$  and  $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$ . In SGD,  $f'$  is usually computed by a small mini-batch of data so that there is an explicit stochasticity in the parameter updates as shown in left of Figure 2.5. The noisy behaviour is able to help the optimization to escape from a local minimum and reach potentially a better solution.

Another type of gradient-based method is based on the use of Hessian, the second order derivatives. It can be understood from the Taylor expansion of the objective function.

$$f(\theta) \approx f(\theta_0) + (\theta - \theta_0)^T f'(\theta_0) + \frac{1}{2}(\theta - \theta_0)^T f''(\theta_0)(\theta - \theta_0) \quad (2.7.3)$$

This type of method tries to establish a local quadratic approximation of the objective function around the current parameter value  $\theta_0$ , as shown in right of Figure 2.5. Compared with SGD, some second order methods require to compute Hessian, or at least to maintain an approximation to it, which is costly. Furthermore, they do not work well when small mini-batches are used, as in SGD.

One desirable property of MLP, including both feed-forward and recurrent neural networks, is that they are universal approximators. The seminal paper from [Stinchcombe and White \(1989\)](#) established that an MLP with one hidden layer and arbitrary squashing function in the hidden layer, linear in the output layer, is capable of approximating any measurable function to any desired degree of accuracy provided enough hidden units. Therefore, changing the number of hidden units indicates a precision by which the function is approximated.

Another desirable property of MLPs is its scalability. When the dimensionality of inputs increases, the number of parameters, or the degree of freedom, only grows linearly. This property stands up when MLPs are compared with non-parametric algorithms whose number of degrees of freedom may need to grow exponentially with the number of inputs. However, this property is obtained at the price of having to deal with a much harder optimization problem and of an assumption on the function to be learned.

On the other hand, training neural networks faces two major difficulties. Firstly, neural networks are highly flexible models and then tend to overfit quickly with insufficient data, hence generalize poorly. To alleviate overfitting, different regularization techniques have been used extensively. The most recent collection of them may be found in the textbook [Goodfellow et al. \(2016\)](#). Secondly, training a deep model with many hidden layers has been quite difficult in general. The main reason of this difficulty, as has been hypothesized by many, is that the optimization problem that learning has to solve becomes harder and harder, either due to bad local minima or ill-conditioning. The existence of saddle points and their impact to optimization have been discussed extensively in [Choromanska et al. \(2015\)](#); [Dauphin et al. \(2014\)](#). To make the optimization even harder, back-propagation may be problematic in very deep neural networks, because as gradients are back-propagated through layers, the error signal is multiplied by the Jacobian (the derivative of outputs w.r.t inputs) of the hidden units layer by layer. Eventually, the error signal may become tiny at the lower layers ([Bengio et al., 1994](#); [Pascanu et al., 2013](#)). As a result, learning in the lower layers is more difficult. There have been successful attempts towards training deep neural networks. For example, [Hinton et al. \(2006b\)](#); [Bengio et al. \(2007\)](#); [Vincent et al. \(2010\)](#) suggested greedy layer-wised unsupervised pre-training to initialize the parameters of a deep MLP. [Glorot and Bengio \(2010\)](#) have proposed a new initialization equation for deep MLP. [Weston et al. \(2008\)](#) uses semi-supervised embeddings to provide extra information to guide a deep MLP. From a pure optimization point of view, carefully designed optimization methods ([Martens, 2010](#); [Martens and Sutskever, 2011](#); [Martens and Grosse, 2015](#); [Grosse and Martens, 2016](#)) have been proved to work well on both feedforward and recurrent neural networks. Nonetheless, the computation is demanding and remains a bottleneck in dealing with large datasets.

### 2.7.2. Sampling in graphical models

Quite often, learning requires the ability to compute the expectation of a function of interest with respect to some random variables. The random variables obey a very complicated distribution so that the only way to compute the expectation is to use samples of those random variables. This is the idea of the Monte Carlo method.

$$\mathbb{E}_x[f(x)] = \int f(x)p(x)dx \approx \frac{1}{N} \sum_{i=1}^N f(x_i) \quad (2.7.4)$$

where one draws samples  $x_i \sim p(x)$ . This is an unbiased estimator. Monte Carlo method is used in the importance sampling technique

$$\mathbb{E}_{x \sim p(x)}[f(x)] = \int f(x)p(x)dx \quad (2.7.5)$$

$$\begin{aligned} &= \int f(x) \frac{p(x)}{q(x)} q(x) dx \\ &= \mathbb{E}_{x \sim q(x)} \left[ f(x) \frac{p(x)}{q(x)} \right] \\ &\approx \frac{1}{N} \sum_{n=1}^N f(x) \frac{p(x)}{q(x)} \end{aligned} \quad (2.7.6)$$

Importance sampling provides a tool to compute expectations of  $f$  with respect to one distribution by drawing samples from another distribution. This is an unbiased estimator.

#### 2.7.2.1. Markov chain Monte Carlo

Importance sampling is less efficient when the dimensionality of the distribution is high. In very high dimensional space, the variance of importance sampling will be too high, making it impractical to use. Markov chain Monte Carlo (MCMC) methods are designed to solve this problem. Given a probability distribution  $p(x)$ , one designs a Markov chain whose stationary distribution is  $p(x)$ . Then sampling from  $p(x)$  amounts to running the Markov chain starting at an arbitrary point  $x_0$ . Designing a Markov chain requires carefully specifying its transition matrix  $T$  whose entry is defined as  $T(x_i|x_j)$ .  $T$  needs to satisfy the following conditions:

- (1) Irreducibility. By using  $T$ , it is possible to reach any state  $i$  from any state  $j$ .
- (2) Aperiodicity. It is impossible to revisit any state in every  $k > 1$  time steps.
- (3) Reversibility/Detailed balance, a sufficient but not necessary condition.  $p(x_i)T(x_j|x_i) = p(x_j)T(x_i|x_j)$

More specifically, in a discrete time homogeneous Markov chain with finite number of states, irreducibility and aperiodicity together guarantee that the chain has a unique stationary distribution  $\pi(x)$ . Reversibility makes sure  $\pi(x) = p(x)$ . In fact, by Perron-Frobenius theorem,  $\pi(x)$  is the eigenvector of  $T$  with eigenvalue 1.

A generalization of Markov chains to an infinite number of states is possible. This is particularly useful when  $\pi(x)$  is continuous. Besides the above 3 conditions,  $T$  has to satisfy recurrence, meaning it is possible to revisit any state, preventing the chain from diverging into infinity.

The Metropolis-Hastings algorithm provides a framework to sample from a complex distribution using Markov chains. At a step  $t$  with a sample  $x^{(t)}$ , the algorithm draws a sample from a proposal distribution  $q(x^{t+1}|x^t)$  and accept  $x^{t+1}$  with the following probability

$$A(x^{t+1}, x^t) = \min(1, \frac{p(x^{t+1})q(x^t|x^{t+1})}{p(x^t)q(x^{t+1}|x^t)}) \quad (2.7.7)$$

In representation learning, MCMC has a great importance. Most MRF models, such as RBMs, DBMs and DBNs require MCMC in the inner loop of learning.





# Chapter 3

---

## LEARNING VISUAL REPRESENTATIONS

### 3.1. THE COMPUTER VISION CHALLENGE

What is “computer vision” in the scope of this thesis? To answer this question, one may be aware of a related field called “computer graphics”, where physical processes are being studied in order to *generate* realistic pictures and movies. Computer vision refers to the inverse process where higher level semantics, such as object’s location and identity, are being computed from the visual inputs. It is sometimes easy to confuse “computer vision” with the classical term “digital image processing” where much emphasis is put on obtaining, transferring and storing images with substantial knowledge from signal processing.

Humans are remarkably adept at visual understanding while computers are far more error-prone. In fact, a human is so good at it that the difficulty of the task is usually underestimated. For instance in 1966, Marvin Minsky at MIT asked an undergraduate student Gerald Jay Sussman to “spend a summer linking a camera to a computer and describe what it saw” (Boden, 2006). And only recently in Krizhevsky et al. (2012b), the computer managed to differentiate objects belonging to 1000 categories with a near-human accuracy.

### 3.2. VISUAL REPRESENTATIONS

It is more and more evident that the fundamental step of tackling any major computer vision challenge such as segmentation, recognition and detection, is to come up with a mathematical construct, often in the form of a scalar or a vector, to represent the object of interest. In fact, the problem becomes much easier to solve if one could represent the inputs in a way such that the direct mapping between the inputs and the outputs becomes immediately apparent.

Finding such a good visual representation poses a fundamental challenge in computer vision due to the complicated physical interaction among objects and their environment. As

a result, the appearance of an object is largely influenced by variations in viewpoint, illumination, scale, occlusion, background clutter, intra-class variations, and camera movements. For instance, in figure 3.1, it is easy to distinguish the blue surgeon from the red clown by representing them with only one scalar – the color. This representation, however, fails to be invariant to the lighting change in the physical environment as any model based solely on such a representation would mistakenly confuse fish (b) and (c) to be of different types. Similarly, (e) would be very likely to require a different representation from (d) for the cat to be successfully recognized from the background.

Furthermore, it is not only laborious but also difficult to program rules that are used to extract representations. For instance, in Figure 3.1, it is not obvious to construct representations that are reflective of emotions in (f) and (g), especially considering both images contain extremely bright and dark pixel intensity.

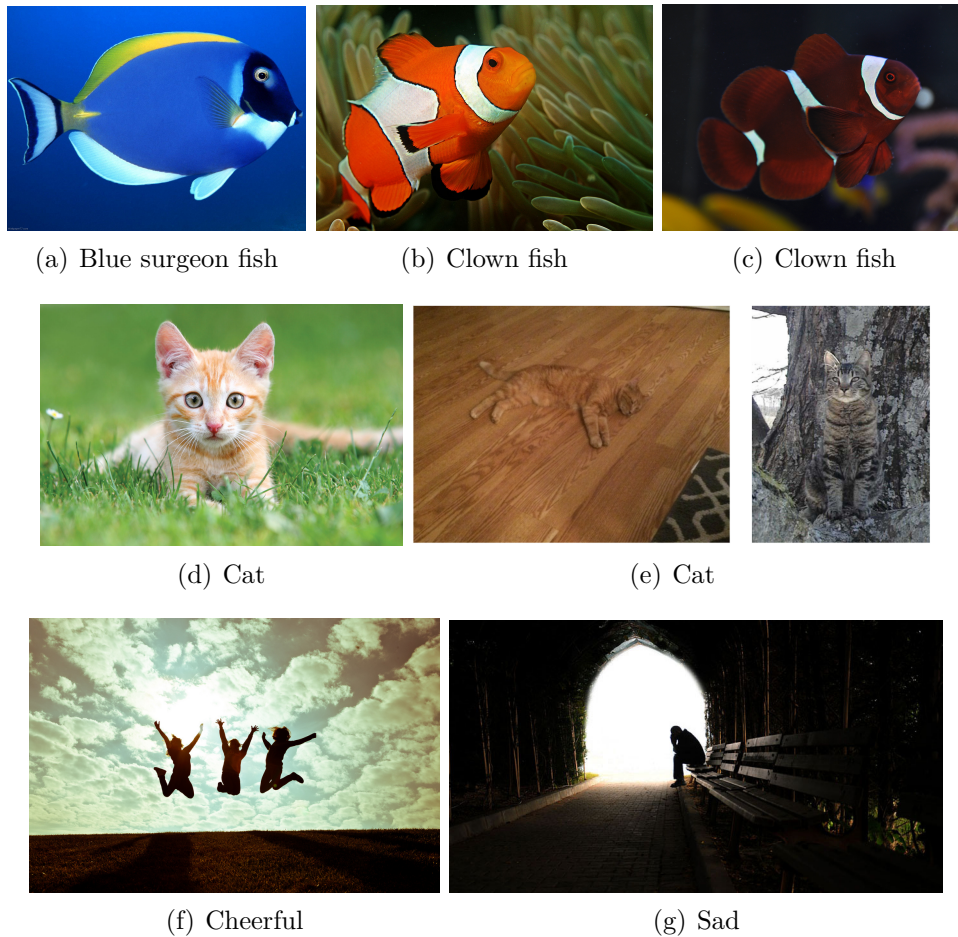


FIG. 3.1. A good representation is invariant to the physical interactions and environmental changes such as (a) – (e). A good representation may not be easily defined with handwritten rules such as (f) and (g).

Therefore, the core of computer vision research lies upon the degree to which such a universal challenge is addressed with respect to a wide range of applications. Admittedly, the approach of designing representations that are tuned to a certain type of application scales poorly with respect to the difficulty of computer vision tasks. A good representation needs to be both computationally and statistically efficient.

To achieve this, traditional approaches rely heavily on domain expertise to hand-craft representations that are typically localized and low-level. On the other hand, a more effective approach is to learn such a semantically rich and high-level representation automatically from raw inputs such as images and video frames. The latter is typically achieved by learning with variants of neural networks with the help of carefully curated datasets. This PhD work contributes directly to the second thread.

### 3.3. TRADITIONAL APPROACHES

#### 3.3.1. Image descriptors and encoding

The research in computer vision has produced a long history of crafting ingenious representations in tackling challenging visual tasks. A brief history can be found in popular textbooks such as Szeliski (2010); Forsyth and Ponce (2003) and more recently Prince (2012). In the course of several decades of research, a commonly used pipeline has emerged and ma-

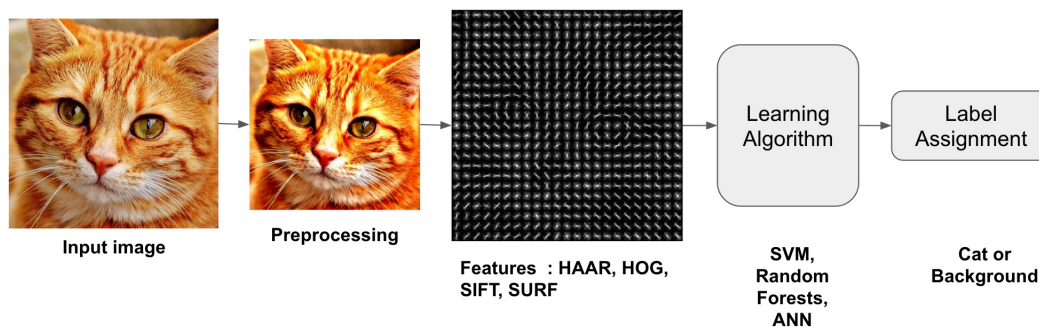


FIG. 3.2. Credit: <http://www.learnopencv.com/>. Classical computer vision approach may be separated into several stages each of which has its particular set of algorithms.

tured under various different application context. Such a “standard” pipeline is demonstrated in Figure 3.2 with the important task of object recognition. Much effort has been devoted to the feature extraction stage between preprocessing and learning. Viola and Jones (2001) popularized the idea of applying simple and small detectors (such as HAAR wavelet) region by region on the raw pixels in a sliding window fashion to detect faces. Following this idea, Dalal and Triggs (2005) proposed HOG filters designed to compute pixel value gradients in

the neighboring area and use the computed gradient map to represent pedestrians. To make the computed feature invariant to geometric and photometric transformation, the map is further evenly divided into small regions each of which is represented by its localized summary statistics in the format of a histogram. Histograms from different regions are further concatenated to yield a representation of the entire image. Aggregation of summary from different regions of an image has become a common theme in feature extraction. Among a large family of techniques for visual feature extraction and description, the SIFT descriptor in [Lowe \(1999\)](#) is among the most popular where additional heuristics are introduced to make the final representation less susceptible to partial occlusion, uniform scaling, orientation and illumination changes.

After feature extraction, the resulting representation is a concatenation of different number of local descriptors. For example, in SIFT, a local descriptor is typically a vector of length 128, and the number of local descriptors varies from hundreds to thousands per image. It is however highly desirable to represent each image by a vector of a fixed length to facilitate the following stage where machine learning models are applied to perform higher-level visual understanding tasks. To accomplish this, [Fei-Fei and Perona \(2005\)](#) borrow the idea of “bag of words” representation of a document from natural language processing and propose a “bag of visual words” for representing an image. The idea is illustrated in Figure 3.3.

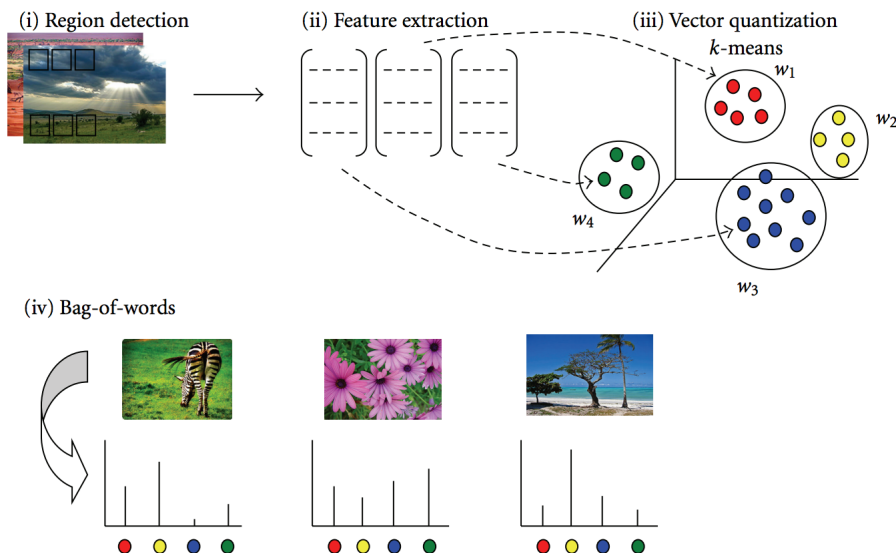


FIG. 3.3. Credit: [\(Tsai, 2012\)](#). Bag of visual words representation for images. Region-based descriptors are first clustered by  $k$ -means in (iii), followed by a histogram where each bin is a cluster, the size of the bin being the number of regional descriptors that fall within that cluster.

The final stage of the classical approach is composed of a learning module that operates on the fixed-size vector. Such modules are usually a nonlinear model with limited model

capacity as in Yang et al. (2009) or a linear model as in Wang et al. (2010). It is important to note that the learning module does not learn the representation, but the mapping from the representations to the targets.

### 3.3.2. Video descriptors and encoding

The classical approaches for video understanding follow a similar path as that for images. Among descriptors designed specifically to capture spatio-temporal patterns, the optical flow is among the most widely studied. Initially proposed by Horn and Schunck (1981); Lucas et al. (1981) to capture movement of pixels across neighboring frames, it has been improved over decades (Anandan, 1989; Sun et al., 2014). The current state-of-the-art action recognition system relies heavily on neural networks trained on both raw pixels and the optical flow as demonstrated in Simonyan and Zisserman (2014a); Ye et al. (2015); Wang et al. (2015). In addition to optical flow, the work of Laptev (2005); Scovanner et al. (2007); Willems et al. (2008) extends the core idea of SIFT to videos by computing spatial-temporal interest points. The work of Klaser et al. (2008) extends HOF by computing pixel gradients in its 3D neighborhood instead of 2D. By far, the best results on video understanding tasks are still dominated by classical approaches such as dense trajectories in Wang et al. (2011); Wang and Schmid (2013); Wang et al. (2015) combined with some variants of neural networks.

The vector is then summarized by  $k$ -means to represent the entire video. This is again followed by a simple classifier to perform high-level video understanding tasks. The standard  $k$ -means encoding used in both 2D and 3D classical pipeline has since been improved by more advanced encoding techniques such as VLAD encoding in Jégou et al. (2010) and Fisher vector encoding in Perronnin et al. (2010).

## 3.4. LEARNING-BASED APPROACHES

### 3.4.1. Supervised learning of visual representation

The classical approach for understanding visual inputs typically incorporates some simple machine learning modules in its pipeline, such as the use of  $k$ -means for aggregating local visual words and SVM for classifying visual words into higher-level concepts such as the object’s identity and location, already illustrated by Figure 3.3 and Figure 3.2. Those learning modules play critical roles in obtaining good performance. It is therefore not entirely surprising that carefully extending learning into other traditional modules in such a pipeline would further improve the performance. There has been plenty of evidence that suggests this direction, especially in the realm of supervised learning with neural networks. After the impressive object recognition results in Krizhevsky et al. (2012b), many challenging problems in computer vision have been reexamined by incorporating neural networks that are learnable. The results are outstanding.



Convolutional neural networks have become the predominant model for object recognition with more refined architectures such as Krizhevsky et al. (2012b); Szegedy et al. (2014); Simonyan and Zisserman (2015); He et al. (2016). Similarly the collected work in object detection in Sermanet et al. (2013); Girshick et al. (2014); Girshick (2015); Ren et al. (2015) has overshadowed the conventional descriptor-based approaches. Models based on neural networks share the same success in solving many other problems such as visual question answering in Antol et al. (2015), semantic segmentation in Long et al. (2015); Ronneberger et al. (2015); Milletari et al. (2016), visual captioning (Xu et al., 2015; Yao et al., 2015a), scene recognition (Farabet et al., 2013; Zhou et al., 2014), action recognition (Simonyan and Zisserman, 2014a; Ji et al., 2013). Chapter 5, 7 and 9 of this thesis also make contributions in the supervised learning setup.

### 3.4.2. Unsupervised learning of visual representation

Unlike supervised learning where the primary goal is to engineer computer vision algorithms to solve challenging tasks, the focus of unsupervised learning of visual representations is more holistic, more far-reaching, and has been largely driven by perhaps one of the most ambitious goals – understanding the visual functionality of the brain. A neural network in this regard is the most natural choice to model the computational mechanism of the visual cortex due to its massively parallel and distributed style to process information. A comprehensive view of neural networks and their connections with computational neuroscience is beyond the scope of this thesis, but may be found in Dayan and Abbott (2003); Trappenberg (2009); Sterratt et al. (2011).

**Learning with linear and shallow neural networks.** From a modelling point of view, the last few decades have produced variants of neural networks that capture spatial patterns in an unsupervised fashion. One fundamental idea for modelling natural images is to represent them with a linear superposition of basic components. This is the case of neural PCA in Baldi and Hornik (1989); Oja (1992) where the inputs are linearly transformed into a subspace so that the resulting components are linearly independent. Similarly, neural ICA algorithms in Karhunen et al. (1997); Hyvärinen and Oja (2000) find a new representation of an input such that its components are as independent as possible. Such a process is realized by linearly projecting inputs into a carefully learned basis. In Ranzato et al. (2007), the idea of sparse coding (Olshausen and Field, 1996) is realized by using an auto-encoder with regularization terms that encourage the sparsity in the linearly encoded representation. Historically, PCA, ICA and sparse coding are solved by using heuristics other than neural networks. It is however the marriage between neural networks and those classical principles that opened the possibility of extending the simple linear transformation used above into nonlinear ones, which has proved to be a fundamental step in learning better visual representations, as argued by Bengio et al. (2009); LeCun et al. (2015); Schmidhuber (2015).

**Learning with nonlinear and deep neural networks.** Going beyond shallow and linear models for unsupervised learning, neural networks share much of their properties with directed graphical models such as Sigmoid Belief Networks in [Saul et al. \(1996\)](#), Variational Auto-encoders in [Kingma and Welling \(2013\)](#), undirected graphical models such as Deep Boltzmann Machines [Salakhutdinov and Hinton \(2009a\)](#) and Deep Belief Networks in [Hinton et al. \(2006a\)](#). Compared with shallow and linear models, these models are typically more difficult to train as training usually contains an inference component that is computationally intractable and therefore requires approximations.

**Recent trend.** In the most recent years, the community has focused much of its attention on developing deep and highly nonlinear models that either have a tractable log-likelihood such as Deep Neural Autoregressive Density Estimator in [Uria et al. \(2013a\)](#), or alternative estimators as in Generative Stochastic Networks in [Bengio et al. \(2014, 2013a\)](#). Most recently, Generative Adversarial Networks ([Goodfellow et al., 2014](#)) have emerged as an important principle to generate natural images as in [Reed et al. \(2016\)](#); [Radford et al. \(2015\)](#), and videos of a few frames as in [Vondrick et al. \(2016\)](#). In addition, Recurrent Neural Networks have found their broad application in generating both small images ([Gregor et al., 2015](#)) and simple videos ([Kalchbrenner et al., 2016](#); [Srivastava et al., 2015a](#)). A key component in all these models is a deep neural network that is capable of capturing complicated spatio-temporal correlation among visual inputs. Such a powerful component is trained by standard backpropagation ([LeCun et al., 2012](#)). Compared with the classical approaches in [Section 5.2.3](#), neural networks are extremely flexible and powerful models that can be trained to perform visual tasks. This, to some extent, reduces the amount of human intervention and most importantly scales well across different types of tasks. It is particularly interesting to see that over the years, other useful priors have emerged as new ways to learn visual representations, including those that exploit the temporal smoothness prior and make frame predictions in [Mathieu et al. \(2016\)](#); [Lotter et al. \(2017\)](#); [Misra et al. \(2016\)](#). Although not directly trained to generate the entire video, these maximum log-likelihood alternatives find their way in learning useful video representations. [Chapter 14](#) discusses those most recent developments at length.





# Chapter 4

---

## PROLOGUE TO FIRST ARTICLE

### 4.1. ARTICLE DETAIL

**Describing videos by exploiting temporal structure.** Yao, Li and Torabi, Atousa and Cho, Kyunghyun and Ballas, Nicolas and Pal, Christopher and Larochelle, Hugo and Courville, Aaron. IEEE international conference on computer vision (ICCV), 2015.

*Personal contribution.* The making of the paper started with Dr. Torabi and I working on separate video projects. The idea of applying the “soft-attention” on video frames became obvious after the success of the image captioning model from [Xu et al. \(2015\)](#). Therefore Prof. Courville, Dr. Ballas, Dr. Cho, Dr. Torabi and I all conceived the similar idea. After Dr. Cho and I implemented it, I conducted experiments on both video captioning datasets used in the paper. I wrote the initial draft with the help of Dr. Cho. The writing was further improved drastically by everyone in the author list. In addition, Dr. Torabi made a significant contribution to the development and training of the 3D Convolutional neural networks, a model that provides frame-wise features to the attention-based LSTM.

### 4.2. CONTEXT

This work explores the possibility of generating natural language descriptions from videos, with the description being a one-sentence English summary of the main content of a 10-30 seconds video snippet. In addition, the progress in using recurrent neural networks (RNNs) for image description has motivated the exploration of their application for video description. However, while images are static, working with videos requires modeling their dynamic temporal structure and then properly integrating that information into a natural language description. In this context, we propose an approach that successfully takes into account both the local and global temporal structure of videos to produce descriptions. First, our approach incorporates a spatial temporal 3-D convolutional neural network (3-D CNN) representation of the short temporal dynamics. The 3-D CNN representation is trained on video action recognition tasks, so as to produce a representation that is tuned to human

motion and behavior. Second we propose a temporal attention mechanism that allows one to go beyond local temporal modeling and *learns* to automatically select the most relevant temporal segments given the text-generating RNN. Our approach exceeded the then state-of-art for both BLEU and METEOR metrics on the Youtube2Text dataset. We also present results on a new, larger and more challenging dataset of paired video and natural language descriptions.

### 4.3. CONTRIBUTIONS

This paper makes the following contributions in the context of bridging the gap of between vision (in the form of videos) and language.

- It identifies two different types of temporal structure in videos, the local one and the global one.
- It is the first paper to capture the local structure by using a 3D ConvNet that starts from locally summarized descriptors instead of raw pixels to reduce the computational complexity. All the previous work focused on training 3D ConvNet from raw pixels.
- It is the first paper to capture the global structure in videos by using soft-attention, extending the previous work that applies mean-pooling on frames.
- By the time of this publication, this paper claimed the state-of-the-art results on Youtube2Text, a widely used benchmark dataset for video captioning. And this paper is among the first to include the result on the much more challenging Movie Description dataset.

# Chapter 5

---

## DESCRIBING VIDEOS BY EXPLOITING TEMPORAL STRUCTURE

### 5.1. INTRODUCTION

The task of automatically describing videos containing rich and open-domain activities poses an important challenges for computer vision and machine learning research. It also has a variety of practical applications. For example, every minute, 100 hours of video are uploaded to YouTube.<sup>1</sup> However, if a video is poorly tagged, its utility is dramatically diminished (Morsillo et al., 2010). Automatic video description generation has the potential to help improve indexing and search quality for online videos. In conjunction with speech synthesis technology, annotating video with natural language descriptions also has the potential to benefit the visually impaired.

While image description generation is already considered a very challenging task, the automatic generation of video description carries additional difficulties. Simply dealing with the sheer quantity of information contained in video data is one such challenge. Moreover, video description involves generating a sentence to characterize a video clip lasting typically 5 to 10 seconds, or 120 to 240 frames. Often such clips contain complex interactions of actors and objects that evolve over time. All together it amounts to a vast quantity of information, and attempting to represent this information using a single, temporally collapsed feature representation is likely to be prone to clutter, with temporally distinct events and objects being potentially fused incoherently. It is therefore important that an automatic video description generator *exploit the temporal structure* underlying video.

We argue that there are two categories of temporal structure present in video: (1) local structure and (2) global structure. Local temporal structure refers to the fine-grained motion information that characterizes punctuated actions such as “answering the telephone” or “standing up”. Actions such as these are relatively localized in time, evolving over only a few consecutive frames. On the other hand, when we refer to global temporal structure in video,

---

<sup>1</sup><https://www.youtube.com/yt/press/statistics.html> accessed on 2015-02-06.



FIG. 5.1. High-level visualization of our approach to video description generation. We incorporate models of both the local temporal dynamic (i.e. within blocks of a few frames) of videos, as well as their global temporal structure. The local structure is modeled using the temporal feature maps of a 3-D CNN, while a temporal attention mechanism is used to combine information across the entire video. For each generated word, the model can focus on different temporal regions in the video. For simplicity, we highlight only the region having the maximum attention above.

we refer to the sequence in which objects, actions, scenes and people, etc. appear in a video. Video description may well be termed video summarization, because we typically look for a single sentence to summarize what can be a rather elaborate sequence of events. Just as good image descriptions often focus on the more salient parts of the image for description, we argue that good video description systems should selectively focus on the most salient features of a video sequence.

Recently, Venugopalan et al. (Venugopalan et al., 2015) used a so-called encoder-decoder neural network framework (Cho et al., 2014) to automatically generate the description of a video clip. They extracted appearance features from each frame of an input video clip using a previously trained convolutional neural network (Krizhevsky et al., 2012b). The features from all the frames, or subsampled frames, were then collapsed via simple averaging to result in a single vector representation of the entire video clip. Due to this indiscriminate averaging of all the frames, this approach risks ignoring much of the temporal structure underlying the video clip. For instance, it is not possible to tell the order of the appearances of two objects from the collapsed features.

In this paper, we introduce a temporal attention mechanism to exploit *global* temporal structure. We also augment the appearance features with action features that encode *local* temporal structure. Our action features are derived from a spatio-temporal convolutional neural network (3-D CNN) (Tran et al., 2014; Karpathy et al., 2014; Ji et al., 2013). The temporal attention mechanism is based on a recently proposed soft-alignment method (Bahdanau et al., 2015) which was used successfully in the context of machine translation. While generating a description, the temporal attention mechanism selectively focuses on a small subset of frames, making it possible for the generator to describe only the objects and/or activities in that subset (see Fig. 9.2 for the graphical illustration). Our 3-D CNN, on the other hand, starts from both temporally and spatially local motion descriptors of video and

hierarchically extracts more abstract action-related features. These features preserve and emphasize important local structure embedded in video for use by the description generator.

We evaluate the effectiveness of the proposed mechanisms for exploiting temporal structure on the most widely used open-domain video description dataset, called the Youtube2Text dataset (Chen and Dolan, 2011a), which consists of 1,970 video clips with multiple descriptions per video. We also test the proposed approaches on a much larger, and more recently proposed, dataset based on the descriptive video service (DVS) tracks in DVD movies (Torabi et al., 2015), which contains 49,000 video clips.

Our work makes the following contributions: 1) We propose the use of a novel 3-D CNN-RNN encoder-decoder architecture which captures local spatio-temporal information. We find that despite the promising results generated by both prior work and our own here using static frame CNN-RNN video description methods, our experiments suggest that it is indeed important to exploit local temporal structure when generating a description of video. 2) We propose the use of an attention mechanism within a CNN-RNN encoder-decoder framework for video description and we demonstrate through our experiments that it allows features obtained through the global analysis of static frames throughout the video to be used more effectively for video description generation. Furthermore, 3) we observe that the improvements brought by exploiting global and local temporal information are complimentary, with the best performance achieved when both the temporal attention mechanism and the 3-D CNN are used together.

## 5.2. VIDEO DESCRIPTION GENERATION USING AN ENCODER-DECODER FRAMEWORK

In this section, we describe a general approach, based purely on neural networks to generate video descriptions. This approach is based on the encoder-decoder framework (Cho et al., 2014), which has been successfully used in machine translation (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015) as well as image caption generation (Kiros et al., 2014a; Donahue et al., 2014; Vinyals et al., 2014; Xu et al., 2015; Karpathy and Fei-Fei, 2014).

### 5.2.1. Encoder-Decoder Framework

The encoder-decoder framework consists of two neural networks; the encoder and the decoder. The encoder network  $\phi$  encodes the input  $\mathbf{x}$  into a continuous-space representation which may be a variable-sized set  $V = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$  of continuous vectors:

$$V = \{\mathbf{v}_1, \dots, \mathbf{v}_n\} = \phi(\mathbf{x}).$$

The architecture choice for the encoder  $\phi$  depends on the type of input. For example, in the case of machine translation, it is natural to use a recurrent neural network (RNN) for the encoder, since the input is a variable-length sequence of symbols (Sutskever et al., 2014; Cho et al., 2014). With an image as input, a convolutional neural network (CNN) is another good alternative (Xu et al., 2015).

The decoder network generates the corresponding output  $y$  from the encoder representation  $V$ . As was the case with the encoder, the decoder’s architecture must be chosen according to the type of the output. When the output is a natural language sentence, which is the case in automatic video description, an RNN is a method of choice.

The decoder RNN  $\psi$  runs sequentially over the output sequence. In brief, to generate an output  $y$ , at each step  $t$  the RNN updates its internal state  $\mathbf{h}_t$  based on its previous internal state  $\mathbf{h}_{t-1}$  as well as the previous output  $y_{t-1}$  and the encoder representation  $V$ , and then outputs a symbol  $y_t$ :

$$\begin{bmatrix} y_t \\ \mathbf{h}_t \end{bmatrix} = \psi(\mathbf{h}_{t-1}, y_{t-1}, V) \quad (5.2.1)$$

where for now we simply note as  $\psi$  the function updating the RNN’s internal state and computing its output. The RNN is run recursively until the end-of-sequence symbol is generated, i.e.,  $y_t = \langle \text{eos} \rangle$ .

In the remaining of this section, we detail choices for the encoder and decoder for a basic automatic video description system, taken from (Venugopalan et al., 2015) and on which our work builds.

### 5.2.2. Encoder: Convolutional Neural Network

Deep convolutional neural networks (CNNs) have recently been successful at large-scale object recognition (Krizhevsky et al., 2012b; Szegedy et al., 2015). Beyond the object recognition task itself, CNNs trained for object recognition have been found to be useful in a variety of other computer vision tasks such as object localization and detection (see, e.g., (Sermanet et al., 2014)). This has opened a door to a flood of computer vision systems that exploit representations from upper or intermediate layers of a CNN as generic high-level features for vision. For instance, the activation of the last fully-connected layer can be used as a fixed-size vector representation (Kiros et al., 2014a), or the feature map of the last convolutional layer can be used as a set of spatial feature vectors (Xu et al., 2015).

In the case where the input is a video clip, an image-trained CNN can be used for each frame separately, resulting in a single vector representation  $\mathbf{v}_i$  of the  $i$ -th frame.

This is the approach proposed by (Venugopalan et al., 2015), which used the convolutional neural network from (Krizhevsky et al., 2012b). In our work here, we will also consider using

the CNN from (Szegedy et al., 2015), which has demonstrated higher performance for object recognition.

### 5.2.3. Decoder: Long Short-Term Memory Network

As discussed earlier, it is natural to use a recurrent neural network (RNN) as a decoder when the output is a natural language sentence. This has been empirically confirmed in the contexts of machine translation (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015), image caption generation (Vinyals et al., 2014; Xu et al., 2015) and video description generation in open (Venugopalan et al., 2015) and closed (Donahue et al., 2014) domains. Among these recently successful applications of the RNN in natural language generation, it is noticeable that most of them (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015; Vinyals et al., 2014; Xu et al., 2015), if not all, used long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997b) or their variant, gated recurrent units (GRU) (Cho et al., 2014). In this paper, we also use a variant of the LSTM units, introduced in (Zaremba et al., 2014), as the decoder.

The LSTM decoder maintains an internal memory state  $\mathbf{c}_t$  in addition to the usual hidden state  $\mathbf{h}_t$  of an RNN (see Eq. (5.2.1)). The hidden state  $\mathbf{h}_t$  is the memory state  $\mathbf{c}_t$  modulated by an output gate:

$$\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{c}_t,$$

where  $\odot$  is an element-wise multiplication. The output gate  $\mathbf{o}_t$  is computed by

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{E}[y_{t-1}] + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{A}_o \varphi_t(V) + \mathbf{b}_o),$$

where  $\sigma$  is the element-wise logistic sigmoid function and  $\varphi_t$  is a time-dependent transformation function on the encoder features.  $\mathbf{W}_o$ ,  $\mathbf{U}_o$ ,  $\mathbf{A}_o$  and  $\mathbf{b}_o$  are, in order, the weight matrices for the input, the previous hidden state, the context from the encoder and the bias.  $\mathbf{E}$  is a word embedding matrix, and we denote by  $\mathbf{E}[y_{t-1}]$  an embedding vector of word  $y_{t-1}$ .

The memory state  $\mathbf{c}_t$  is computed as a weighted sum between the previous memory state  $\mathbf{c}_{t-1}$  and the new memory content update  $\tilde{\mathbf{c}}_t$ :

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t,$$

where the coefficients – called forget and input gates respectively – are given by

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{E}[y_{t-1}] + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{A}_f \varphi_t(V) + \mathbf{b}_f),$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{E}[y_{t-1}] + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{A}_i \varphi_t(V) + \mathbf{b}_i).$$

The updated memory content  $\tilde{\mathbf{c}}_t$  also depends on the current input  $y_{t-1}$ , previous hidden state  $\mathbf{h}_{t-1}$  and the features from the encoder representation  $\varphi_t(V)$ :

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{E}[y_{t-1}] + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{A}_c \varphi_t(V) + \mathbf{b}_c).$$

Once the new hidden state  $\mathbf{h}_t$  is computed, a probability distribution over the set of possible words is obtained using a single hidden layer neural network

$$\mathbf{p}_t = \text{softmax}(\mathbf{U}_p \tanh(\mathbf{W}_p [\mathbf{h}_t, \varphi_t(V), \mathbf{E}[y_{t-1}]] + \mathbf{b}_p) + \mathbf{d}), \quad (5.2.2)$$

where  $\mathbf{W}_p, \mathbf{U}_p, \mathbf{b}_p, \mathbf{d}$  are the parameters of this network,  $[\dots]$  denotes vector concatenation. The softmax function allows us to interpret  $\mathbf{p}_t$  as the probabilities of the distribution  $p(y_t | y_{<t}, V)$  over words.

At a higher level, the LSTM decoder can be written down as

$$\begin{bmatrix} p(y_t | y_{<t}, V) \\ \mathbf{h}_t \\ \mathbf{c}_t \end{bmatrix} = \psi(\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, y_{t-1}, V). \quad (5.2.3)$$

It is then trivial to generate a sentence from the LSTM decoder. For instance, one can recursively evaluate  $\psi$  and sample from the returned  $p(y_t | \dots)$  until the sampled  $y_t$  is the end-of-sequence symbol. One can also approximately find the sentence with the highest probability by using a simple beam search (Sutskever et al., 2014).

In (Venugopalan et al., 2015), Venugopalan et al. used this type of LSTM decoder for automatic video description generation. However, in their work the feature transformation function  $\varphi_t$  consisted in a simple averaging, i.e.,

$$\varphi_t(V) = \frac{1}{n} \sum_{i=1}^n \mathbf{v}_i, \quad (5.2.4)$$

where the  $v_i$ 's are the elements of the set  $V$  returned by the CNN encoder from Sec. 5.2.2. This averaging effectively collapses all the frames, indiscriminate of their temporal relationships, leading to the loss of temporal structure underlying the input video.

### 5.3. EXPLOITING TEMPORAL STRUCTURE IN VIDEO DESCRIPTION GENERATION

In this section, we delve into the main contributions of this paper and propose an approach for exploiting both the local and global temporal structure in automatic video description.

#### 5.3.1. Exploiting Local Structure: A Spatio-Temporal Convolutional Neural Net

We propose to model the local temporal structure of videos at the level of the temporal features  $V = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$  that are extracted by the encoder. Specifically, we propose to use



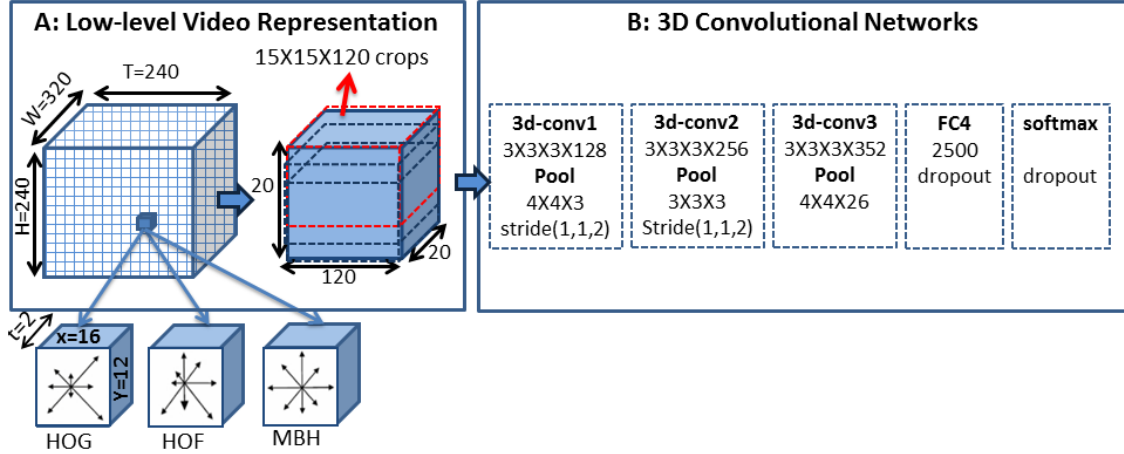


FIG. 5.2. Illustration of the spatio-temporal convolutional neural network (3-D CNN). This network is trained for activity recognition. Then, only the convolutional layers are involved when generating video descriptions.

a spatio-temporal convolutional neural network (3-D CNN) which has recently been demonstrated to capture well the temporal dynamics in video clips (Tran et al., 2014; Karpathy et al., 2014).

We use a 3-D CNN to build the higher-level representations that preserve and summarize the local motion descriptors of short frame sequences. This is done by first dividing the input video clip into a 3-D spatio-temporal grid of  $16 \times 12 \times 2$  (width  $\times$  height  $\times$  timesteps) cuboids. Each cuboid is represented by concatenating the histograms of oriented gradients, oriented flow and motion boundary (HoG, HoF, MbH) (Dalal et al., 2006; Wang et al., 2009) with 33 bins. This transformation is done in order to make sure that local temporal structure (motion features) are well extracted and to reduce the computation of the subsequence 3-D CNN.

Our 3-D CNN architecture is composed of three 3-D convolutional layer, each followed by rectified linear activations (ReLU) and local max-pooling. From the activation of the last 3-D convolution+ReLU+pooling layer, which preserves the temporal arrangement of the input video and abstracts the local motion features, we can obtain a set of temporal feature vectors by max-pooling along the spatial dimensions (width and height) to get feature vectors that each summarize the content over short frame sequences within the video. Finally, these feature vectors are combined, by concatenation, with the image features extracted from single frames taken at similar positions across the video. Fig. 5.2 illustrates the complete architecture of the described 3-D CNN. Similarly to the object recognition trained CNN (see Sec. 5.2.2), the 3-D CNN is pre-train on activity recognition datasets.

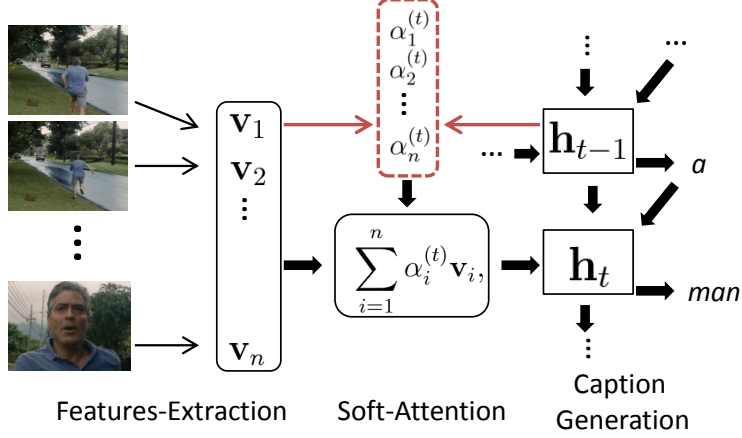


FIG. 5.3. Illustration of the proposed temporal attention mechanism in the LSTM decoder

### 5.3.2. Exploiting Global Structure: A Temporal Attention Mechanism

The 3-D CNN features of the previous section allows us to better represent short-duration actions in a subset of consecutive frames. However, representing a complete video by averaging these local temporal features as in Eq. 5.2.4 would jeopardize the model’s ability to exploit the video’s global temporal structure.

Our approach to exploiting such non-local temporal structure is to let the decoder selectively focus on only a small subset of frames at a time. By considering subsets of frames in sequence, the model can exploit the temporal ordering of objects and actions across the entire video clip and avoid conflating temporally disparate events. Our approach also has the potential of allowing the model to focus on key elements of the video that may have short duration. Methods that collapse the temporal structure risk overwhelming these short duration elements.

Specifically, we propose to adapt the recently proposed soft attention mechanism from (Bahdanau et al., 2015), which allows the decoder to weight each temporal feature vector  $V = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ . This approach has been used successfully by Xu et al. (Xu et al., 2015) for exploiting spatial structure underlying an image. Here, we thus adapt it to exploit the temporal structure of video instead.

Instead of a simple averaging strategy (as shown in Eq. (5.2.4)), we take the *dynamic* weighted sum of the temporal feature vectors such that

$$\varphi_t(V) = \sum_{i=1}^n \alpha_i^{(t)} \mathbf{v}_i,$$

where  $\sum_{i=1}^n \alpha_i^{(t)} = 1$  and  $\alpha_i^{(t)}$ ’s are computed at each time step  $t$  inside the LSTM decoder (see Sec. 5.2.3). We refer to  $\alpha_i^{(t)}$  as the attention weights at time  $t$ .

The attention weight  $\alpha_i^{(t)}$  reflects the relevance of the  $i$ -th temporal feature in the input video given all the previously generated words, i.e.,  $y_1, \dots, y_{t-1}$ . Hence, we design a function that takes as input the previous hidden state  $\mathbf{h}_{t-1}$  of the LSTM decoder, which summarizes all the previously generated words, and the feature vector of the  $i$ -th temporal feature and returns the unnormalized relevance score  $e_i^{(t)}$ :

$$e_i^{(t)} = \mathbf{w}^\top \tanh(\mathbf{W}_a \mathbf{h}_{t-1} + \mathbf{U}_a \mathbf{v}_i + \mathbf{b}_a),$$

where  $\mathbf{w}$ ,  $\mathbf{W}_a$ ,  $\mathbf{U}_a$  and  $\mathbf{b}_a$  are the parameters that are estimated together with all the other parameters of the encoder and decoder networks.

Once the relevance scores  $e_i^{(t)}$  for all the frames  $i = 1, \dots, n$  are computed, we normalize them to obtain the  $\alpha_i^{(t)}$ 's:

$$\alpha_i^{(t)} = \exp\{e_i^{(t)}\} / \sum_{j=1}^n \exp\{e_j^{(t)}\}.$$

We refer to the *attention mechanism* as this whole process of computing the unnormalized relevance scores and normalizing them to obtain the attention weights.

The attention mechanism allows the decoder to selectively focus on only a subset of frames by increasing the attention weights of the corresponding temporal feature. However, we do not explicitly force this type of selective attention to happen. Rather, this inclusion of the attention mechanism enables the decoder to exploit the temporal structure, *if* there is useful temporal structure in the data. Later in Sec. 5.5, we empirically show that this is indeed the case. See Fig. 5.3 for the graphical illustration of the temporal attention mechanism.

## 5.4. RELATED WORK

Video description generation has been investigated and studied in other work, such as (Kojima et al., 2002; Barbu et al., 2012; Rohrbach et al., 2013). Most of these examples have, however, constrained the domain of videos as well as the activities and objects embedded in the video clips. Furthermore, they tend to rely on hand-crafted visual representations of the video, to which template-based or shallow statistical machine translation approaches were applied. In contrast, the approach we take and propose in this paper aims at open-domain video description generation with deep trainable models starting from low-level video representations, including raw pixel intensities (see Sec. 5.2.2) and local motion features (see Sec. 5.3.1).

In this sense, the approach we use here is more closely related to the recently introduced static image caption generation approaches based mainly on neural networks (Kiros et al., 2014a; Donahue et al., 2014; Vinyals et al., 2014; Xu et al., 2015; Karpathy and Fei-Fei, 2014). A neural approach to static image caption generation has recently been applied to video description generation by Venugopalan et al. (Venugopalan et al., 2015). However,

their direct adaptation of the underlying static image caption generation mechanism to the videos is limited by the fact that the model tends to ignore the temporal structure of the underlying video. Such structure has demonstrated to be helpful in the context of event and action classification (Tang et al., 2012; Gaidon et al., 2013; Bojanowski et al., 2014), and is explored in this paper. Other recent work (Rohrbach et al., 2015a) has explored the use of DVS annotated video for video description research and has underscored the observation that DVS descriptions are typically much more relevant and accurate descriptions of the visual content of a video compared to movie scripts. They present results using both DVS and script based annotations as well as cooking activities.

While other work has explored 3-D Deep Networks for video (Taylor et al., 2010; Ji et al., 2013; Karpathy and Fei-Fei, 2014; Simonyan and Zisserman, 2014a) our particular approach differs in a number of ways from prior work in that it is based on CNNs as opposed to other 3-D deep architectures and we focus on pre-training the model on a number of widely used action recognition datasets. In contrast to other 3-D CNN formulations, the input to our 3-D CNN consists of features derived from a number of state of the art image descriptors. Our model is also fully 3-D in that we model entire volumes across a video clip.

In this paper, we use a state-of-the-art static convolutional neural network (CNN) and a novel spatio-temporal 3-D CNN to model input video clips. This way of modeling video using feedforward convolutional neural networks, has become increasingly popular recently (Venugopalan et al., 2015; Simonyan and Zisserman, 2014a; Tran et al., 2014). However, there has also been a stream of research on using recurrent neural networks (RNN) for modeling video clips. For instance, in (Srivastava et al., 2015a), Srivastava et al. propose to use long short-term memory units to extract video features. Ranzato et al. in (Ranzato et al., 2014) also models a video clip with an RNN, however, after vector-quantizing image patches of the video clip. In contrast to other approaches such as (Donahue et al., 2014), which have explored CNN-RNN coupled models for video description, here we use an attention mechanism, use a 3-D CNN and focus on open-domain video description.

## 5.5. EXPERIMENTS

We test the proposed approaches on two video-description corpora: Youtube2Text (Chen and Dolan, 2011a) and DVS (Torabi et al., 2015). Implementations are available at <https://github.com/yaoli/arctic-capgen-vid>.

### 5.5.1. Datasets

#### 5.5.1.1. Youtube2Text

The Youtube2Text video corpus (Chen and Dolan, 2011a) is well suited for training and evaluating an automatic video description generation model. The dataset has 1,970

video clips with multiple natural language descriptions for each video clip. In total, the dataset consists of approximately 80,000 video / description pairs, with the vocabulary of approximately 16,000 unique words. The dataset is open-domain and covers a wide range of topics including sports, animals and music. Following (Venugopalan et al., 2015), we split the dataset into a training set of 1,200 video clips, a validation set of 100 clips and a test set consisting of the remaining clips.

#### 5.5.1.2. DVS

The DVS dataset was recently introduced in (Torabi et al., 2015) with a much larger number of video clips and accompanying descriptions than the existing video/description corpora such as Youtube2Text. It contains video clips extracted from 92 DVD movies along with semi-automatically transcribed descriptive video service (DVS) narrations. The dataset consists of 49,000 video clips covering a wide variety of situations. We follow the standard split of the dataset into a training set of 39,000 clips, a validation set of 5,000 clips and a test set of 5,000 clips, as suggested by (Torabi et al., 2015).

### 5.5.2. Description Preprocessing

We preprocess the descriptions in both the Youtube2Text and DVS datasets with “word-punct\_tokenizer” from the NLTK toolbox.<sup>2</sup> We did not do any other preprocessing such as lowercasing and rare word elimination. After preprocessing, the numbers of unique words were 15,903 for Youtube2Text and 17,609 for DVS Dataset.

### 5.5.3. Video Preprocessing

To reduce the computational and memory requirement, we only consider the first 240 frames of each video <sup>3</sup> For appearance features, (trained) 2-D *GoogLeNet* (Szegedy et al., 2015) CNN is used to extract fixed-length representation (with the help of the popular implementation in Caffe (Jia et al., 2014)). Features are extracted from the pool5/7x7\_s1 layer. We select 26 equally-spaced frames out of the first 240 from each video and feed them into the CNN to obtain a 1024 dimensional frame-wise feature vector. We also apply the spatio-temporal 3-D CNN (trained as described in Sec. 5.5.4.2) in order to extract local motion information<sup>4</sup>. When using 3-D CNN without temporal attention, we simply use the 2500-dimensional activation of the last fully-connection layer. When we combine the 3-D CNN with the temporal attention mechanism, we leverage the last convolutional layer

<sup>2</sup> <http://s/www.nltk.org/index.html>

<sup>3</sup> When the video clip has less than 240 frames, we pad the video with all-zero frames to make it into 240-frame long.

<sup>4</sup> We perturb each video along three axes to form random crops by taking multiple  $15 \times 15 \times 120$  cuboids out of the original  $20 \times 20 \times 120$  cuboids, and the final representation is the average of the representations from these perturbed video clips.

representation leading to 26 feature vectors of size 352. Those vector are contatenated with the 2D CNN features resulting in 26 feature vectors with 1376 elements.

TAB. 5. I. Performance of different variants of the model on the Youtube2Text

Model	Youtube2Text			
	BLEU	METEOR	CIDEr	Perp.
Enc-Dec (Basic)	0.3869	0.2868	0.4478	33.09
+ Local (3-D CNN)	0.3875	0.2832	0.5087	33.42
+ Global (Temporal Attention)	0.4028	0.2900	0.4801	27.89
+ Local + Global	<b>0.4192</b>	<b>0.2960</b>	<b>0.5167</b>	<b>27.55</b>
Venugopalan <i>et al.</i> (Venugopalan et al., 2015)	0.3119	0.2687	-	-
+ Extra Data (Flickr30k, COCO)	0.3329	0.2907	-	-
Thomason <i>et al.</i> (Thomason et al., 2014)	0.1368	0.2390	-	-

TAB. 5. II. Performance of different variants of the model on the DVS

Model	DVS			
	BLEU	METEOR	CIDEr	Perp.
Enc-Dec (Basic)	0.003	0.044	0.044	88.28
+ Local (3-D CNN)	0.004	0.051	0.050	84.41
+ Global (Temporal Attention)	0.003	0.040	0.047	66.63
+ Local + Global	<b>0.007</b>	<b>0.057</b>	<b>0.061</b>	<b>65.44</b>

#### 5.5.4. Experimental Setup

##### 5.5.4.1. Models

We test four different model variations for video description generation based on the underlying encoder-decoder framework, with results presented in Table 5. I and Table 5. II. *Enc-Dec (Basic)* denotes a baseline incorporating neither local nor global temporal structure. Is it based on an encoder using the 2-D GoogLeNet CNN (Szegedy et al., 2015) as discussed in Section 5.2.2 and the LSTM-based decoder outlined in Section 5.2.3. *Enc-Dec + Local* incorporates local temporal structure via the integration of our proposed 3-D CNN features (as outlined in Section 5.3.1) with the 2-D GoogLeNet CNN features as described above. *Enc-Dec + Global* adds the temporal attention mechanism of Section 5.3.2. Finally, *Enc-Dec + Local + Global* incorporates both the 3-D CNN and the temporal attention mechanism into the model.

All models otherwise use the same number of temporal features  $\mathbf{v}_i$ . These experiments will allow us to investigate whether the contributions from the proposed approaches are complimentary and can be combined to further improve performance.

#### 5.5.4.2. Training

For all video description generation models, we estimated the parameters by maximizing the log-likelihood:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N \sum_{i=1}^{t_n} \log p(y_i^n | y_{<i}^n, \mathbf{x}^n, \boldsymbol{\theta}),$$

where there are  $N$  training video-description pairs  $(\mathbf{x}^n, y^n)$ , and each description  $y^n$  is  $t_n$  words long.

We used Adadelta (Zeiler, 2012) with the gradient computed by the backpropagation algorithm. We optimized the hyperparameters (e.g. number of LSTM units and the word embedding dimensionality) using random search to maximize the log-probability of the validation set.<sup>5</sup> Training continued until the validation log-probability stopped increasing for 5,000 updates. As mentioned earlier in Sec. 5.3.1, the 3-D CNN was trained on activity recognition datasets. Due to space limitation, details regarding the training and evaluation of the 3-D CNN on activity recognition datasets are provided in the Supplementary Material.

#### 5.5.4.3. Evaluation

We report the performance of our proposed method using test set perplexity and three model-free automatic evaluation metrics. These are BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014) and CIDEr (Vedantam et al., 2014). We use the evaluation script prepared and introduced in (Chen et al., 2015).

### 5.5.5. Quantitative Analysis

In the first block of Table 5. I and Table 5. II, we present the performance of the four different variants of the model using all four metrics: BLEU, METEOR, CIDEr and perplexity. Subsequent lines in the table give comparisons with prior work. The first three rows (Enc-Dec (Basic), +Local and +Global), show that it is generally beneficial to exploit some type of temporal structure underlying the video. Although this benefit is most evident with perplexity (especially with the temporal attention mechanism exploiting global temporal structure), we observe a similar trend with the other model-free metrics and across both Youtube2Text and DVS datasets.

We observe, however, that the biggest gain can be achieved by letting the model exploit *both* local and global temporal structure (the fourth row in Table 5. II and 5. I). We observed this gain consistently across both datasets as well as using all four automatic evaluation metrics.

---

<sup>5</sup> Refer to the Supplementary Material for the selected hyperparameters.



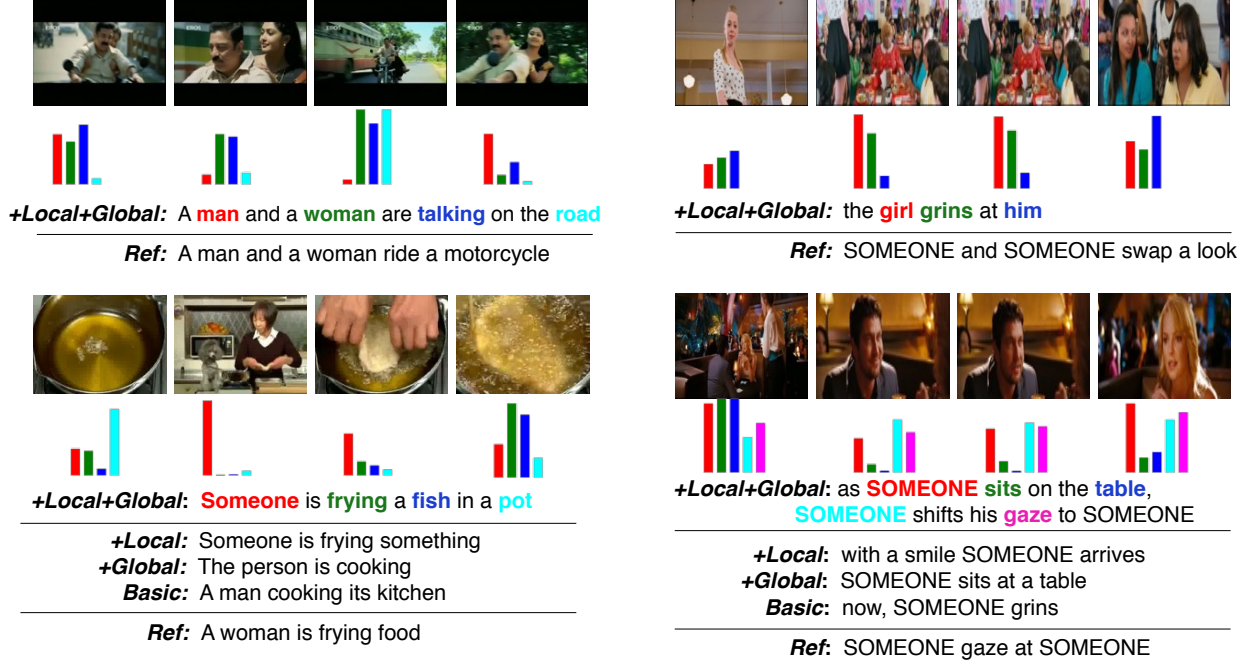


FIG. 5.4. Four sample videos and their corresponding generated and ground-truth descriptions from Youtube2Text (Left Column) and DVS (Right Column). The bar plot under each frame corresponds to the attention weight  $\alpha_i^t$  for the frame when the corresponding word (color-coded) was generated. From the top left panel, we can see that when the word “road” is about to be generated, the model focuses highly on the third frame where the road is clearly visible. Similarly, on the bottom left panel, we can see that the model attends to the second frame when it was about to generate the word “Someone”. The bottom row includes alternate descriptions generated by the other model variations.

### 5.5.6. Qualitative Analysis

Although the model-free evaluation metrics such as the ones we used in this paper (BLEU, METEOR, CIDEr) were designed to reflect the agreement level between reference and generated descriptions, it is not intuitively clear how well those numbers (see Table 5. II and 5. I) reflect the quality of the actual generated descriptions. Therefore, we present some of the video clips and their corresponding descriptions, both generated and reference, from the test set of each dataset. Unless otherwise labeled, the visualizations in this section are from the best model which exploits both global and local temporal structure (the fourth row of Table 5. II and 5. I).

In Fig. 5.4, two video clips from the test set of Youtube2Text are shown. We can clearly see that the generated descriptions correspond well with the video clips. In Fig. 5.4, we show also two sample video clips from the DVS dataset. Clearly, the model does not perform as well on the DVS dataset as it did on Youtube2Text, which was already evident from the



quantitative analysis in Sec. 5.5.5. However, we still observe that the model often focuses correctly on a subset of frames according to the word to be generated. For instance, in the left pane, when the model is about to generate the second “SOMEONE”, it focuses mostly on the first frame. Also, on the right panel, the model correctly attends to the second frame when the word “types” is about to be generated. As for the 3-D CNN local temporal features, we see that they allowed to correctly identify the action as “frying”, as opposed to simply “cooking”.

More samples of the video clips and the generated/reference descriptions can be found in the Supplementary Material, including visualizations from the global temporal attention model alone (see the third row in Table 5. I and Table 5. II).

## 5.6. CONCLUSION

In this work, we address the challenging problem of producing natural language descriptions of videos. We identify and underscore the importance of capturing both local and global temporal structure in addition to frame-wise appearance information. To this end, we propose a novel 3-D convolutional neural network that is designed to capture local fine-grained motion information from consecutive frames. In order to capture global temporal structure, we propose the use of a temporal attentional mechanism that learns the ability to focus on subsets of frames.

Finally, the two proposed approaches fit naturally together into an encoder-decoder neural video caption generator.

We have empirically validated each approach on both Youtube2Text and DVS datasets on four standard evaluation metrics. Experiments indicate that models using either approach improve over the baseline model. Furthermore, combining the two approaches gives the best performance. In fact, we achieved the state-of-the-art results on Youtube2Text with the combination.

Given the challenging nature of the task, we hypothesize that the performance on the DVS dataset could be significantly improved by incorporating another recently proposed dataset (Rohrbach et al., 2015a) similar to the DVS data used here. In addition, we have some preliminary experimental results that indicate that further performance gains are possible by leveraging image caption generation datasets such as MS COCO (Chen et al., 2015) and Flickr (Hodosh et al., 2013). We intend to more fully explore this direction in future work.

## ACKNOWLEDGMENTS

The authors would like to thank the developers of Theano (Bergstra et al., 2010; Bastien et al., 2012). We acknowledge the support of the following organizations for research funding

and computing support: NSERC, FQRNT, Samsung, Calcul Quebec, Compute Canada, the Canada Research Chairs and CIFAR.

# Chapter 6

---

## PROLOGUE TO SECOND ARTICLE

### 6.1. ARTICLE DETAIL

**Oracle performance for visual captioning.** Yao, Li and Ballas, Nicolas and Cho, Kyunghyun and Smith, John R and Bengio, Yoshua. *British Machine Vision Conference (BMVC)*, 2015.

*Personal contributions.* The main idea of establishing the oracle performance was conceived by me during my internship in IBM Watson Research. I wrote all the implementation and most of the paper. Dr. Ballas suggested the experiments with corrupted atoms in the experiment section. The writing of the paper received advice from everyone on the author list.

### 6.2. CONTEXT

The work in Chapter 5, among others around the same time (see references in Section. 7.2.1), piqued the interest in the research community of video captioning. This work continues in this direction and study the broader concept of visual captioning that includes both image and video captioning. It looks at the problem in a different angle and investigates the following two hypotheses:

- The existing video captioning models perform well mainly due to the capacity of modeling language.
- The performance of visual captioning models may be greatly improved by utilizing better visual representations.

In fact, associating images and videos with a natural language description has attracted a great amount of attention recently. The state-of-the-art results on some of the standard datasets have been pushed into the regime where it has become more and more difficult to make significant improvements. Instead of proposing new models, this work investigates performances that an oracle can obtain. In order to disentangle the contribution of the visual model from the language model, our oracle assumes that a high-quality visual concept

extractor is available and focuses only on the language part. We demonstrate the construction of such oracles on MS-COCO, YouTube2Text and LSMDC (a combination of M-VAD and MPII-MD). Surprisingly, despite the simplicity of the model and the training procedure, we show that current state-of-the-art models fall short when being compared with the learned oracle. Furthermore, it suggests the inability of current models in capturing important visual concepts in captioning tasks.

### 6.3. CONTRIBUTIONS

Before this publication, there had not been a systematic study of an empirical performance upper bound on visual captioning systems. This work proposed to view any existing system in two stages: visual atom (see definition in the paper) extraction from videos followed by description generation from atoms. With this assumption, we are able to establish an optimal performance by assuming the first stage is perfect. Such an oracle is used to upper bound the current state of the art models. By comparing the performance with such an oracle, one could quantitatively measure the effectiveness of any visual captioning system. The experiments conducted in the paper further validate that both hypotheses are plausible.

# Chapter 7

---

## ORACLE PERFORMANCE FOR VISUAL CAPTIONING

### 7.1. INTRODUCTION

With standard datasets publicly available, such as COCO and Flickr (Lin et al., 2014; Hodosh et al., 2013; Young, Lai, Hodosh, and Hockenmaier, Young et al.) in image captioning, and YouTube2Text, MVAD and MPI-MD (Guadarrama et al., 2013; Torabi et al., 2015; Rohrbach et al., 2015a) in video captioning, the field has been progressing in an astonishing speed. For instance, the state-of-the-art results on COCO image captioning has been improved rapidly from 0.17 to 0.31 in BLEU (Kiros et al., 2014b; Devlin et al., 2015; Donahue et al., 2014; Vinyals et al., 2014; Xu et al., 2015; Mao et al., 2015; Karpathy and Fei-Fei, 2014; Bengio et al., 2015; Qi Wu et al., 2015). Similarly, the benchmark on YouTube2Text has been repeatedly pushed from 0.31 to 0.50 in BLEU score (Rohrbach et al., 2013; Venugopalan et al., 2015; Yao et al., 2015b; Venugopalan et al., 2015; Xu et al., 2015; Rohrbach et al., 2015; Yu et al., 2015; Ballas et al., 2016).

While obtaining encouraging results, captioning approaches involve large networks, usually leveraging convolution network for the visual part and recurrent network for the language side. It therefore results model with a certain complexity where the contribution of the different component is not clear.

Instead of proposing better models, the main objective of this work is to develop a method that offers a deeper insight of the strength and the weakness of popular visual captioning models. In particular, we propose a trainable oracle that disentangles the contribution of the visual model from the language model.

To obtain such oracle, we follow the assumption that the image and video captioning task may be solved with two steps (Rohrbach et al., 2013; Fang et al., 2015) . Consider the model  $P(\mathbf{w}|\mathbf{v})$  where  $\mathbf{v}$  refers to usually high dimensional visual inputs, such as representations of an image or a video, and  $\mathbf{w}$  refers to a caption, usually a sentence of natural language description. In order to work well,  $P(\mathbf{w}|\mathbf{v})$  needs to form higher level visual concept, either

explicitly or implicitly, based on  $\mathbf{v}$  in the first step, denoted as  $P(\mathbf{a}|\mathbf{v})$ , followed by a language model that transforms visual concept into a legitimate sentence, denoted by  $P(\mathbf{w}|\mathbf{a})$ .  $\mathbf{a}$  refers to *atoms* that are visually perceivable from  $\mathbf{v}$ .

The above assumption suggests an alternative way to build an oracle. In particular, we assume the first step is *close to perfect* in the sense that visual concept (or hints) is observed with almost 100% accuracy. And then we train the best language model conditioned on hints to produce captions.

Using the proposed oracle, we compare the current state-of-the-art models against it, which helps to quantify their capacity of visual modeling, a major weakness, apart from the strong language modeling. In addition, when being applied on different datasets, the oracle offers insight on the intrinsic difficulty and blessing of them, a general guideline when designing new algorithms and developing new models. Finally, we also relax the assumption to investigate the case where visual concept may not be realistically predicted with 100% accuracy and demonstrate a quantity-accuracy trade-off in solving visual captioning tasks.

## 7.2. RELATED WORK

### 7.2.1. Visual captioning

The problem of image captioning has attracted a great amount of attention lately. Early work focused on constructing linguistic templates or syntactic trees based on a set of concept from visual inputs (Kuznetsova et al., 2012; Mitchell et al., 2012; Kulkarni et al., 2013). Another popular approach is based on caption retrieval in the embedding space such as Kiros et al. (2014b); Devlin et al. (2015). Most recently, the use of language models conditioned on visual inputs have been widely studied in the work of Fang et al. (2015) where a maximum entropy language model is used and in Donahue et al. (2014); Vinyals et al. (2014); Xu et al. (2015); Mao et al. (2015); Karpathy and Fei-Fei (2014) where recurrent neural network based models are built to generate natural language descriptions. The work of Devlin et al. (2015) suggests to combine both types of language models. Furthermore, CIDEr (Vedantam et al., 2014) was proposed as an alternative evaluation metric for image captioning and is shown to be more advantageous compared with BLEU and METEOR. To further improve the performance, Bengio et al. (2015) suggests a simple sampling algorithm during training, which was one of the winning recipes for MSR-COCO Captioning challenge<sup>1</sup>, and Jia et al. (2015) suggests the use of extra semantic information to guide the language generation process.

Similarly, video captioning has made substantial progress recently. Early models such as Kojima et al. (2002); Barbu et al. (2012); Rohrbach et al. (2013) tend to focus on constrained domains with limited appearance of activities and objects in videos. They also rely heavily

<sup>1</sup><http://mscoco.org>

on hand-crafted video features, followed by a template-based or shallow statistical machine translation approaches to produce captions. Borrowing success from image captioning, recent models such as Venugopalan et al. (2015); Donahue et al. (2014); Yao et al. (2015b); Venugopalan et al. (2015); Xu et al. (2015); Rohrbach et al. (2015); Yu et al. (2015) and most recently Ballas et al. (2016) have adopted a more general encoder-decoder approach with end-to-end parameter tuning. Videos are input into a specific variant of encoding neural networks to form a higher level visual summary, followed by a caption decoder by recurrent neural networks. Training such type of models are possible with the availability of three relatively large scale datasets, one collected from YouTube by Guadarrama et al. (2013), the other two constructed based on Descriptive Video Service (DVS) on movies by Torabi et al. (2015) and Rohrbach et al. (2015a). The latter two have recently been combined as the official dataset for Large Scale Movie Description Challenge (LSMDC) <sup>2</sup>.

### 7.2.2. Capturing higher-level visual concept

The idea of using intermediate visual concept to guide the caption generation has been discussed in Qi Wu et al. (2015) in the context of image captioning and in Rohrbach et al. (2015) for video captioning. Both work trained classifiers on a predefined set of visual concepts, extracted from captions using heuristics from linguistics and natural language processing. Our work resembles both of them in the sense that we also extract similar constituents from captions. The purpose of this study, however, is different. By assuming perfect classifiers on those visual atoms, we are able to establish the performance upper bounds for a particular dataset. Note that a simple bound is suggested by Rohrbach et al. (2015) where METEOR is measured on all the training captions against a particular test caption. The largest score is picked as the upper bound. As a comparison, our approach constructs a series of oracles that are trained to generate captions given different number of visual hints. Therefore, such bounds are clear indication of models’ ability of capturing concept within images and videos when performing caption generation, instead of the one suggested by Rohrbach et al. (2015) that performs caption retrieval.

## 7.3. ORACLE MODEL

The construction of the oracle is inspired by the observation that  $P(\mathbf{w}|\mathbf{v}) = \sum_{\mathbf{a}} P_{\theta}(\mathbf{w}|\mathbf{a})P(\mathbf{a}|\mathbf{v})$  where  $\mathbf{w} = \{\mathbf{w}_1, \dots, \mathbf{w}_t\}$  denotes a caption containing a sequence of words having a length  $t$ .  $\mathbf{v}$  denotes the visual inputs such as an image or a video.  $\mathbf{a}$  denotes visual concepts which we call “atoms”. We have explicitly factorized the captioning model  $P(\mathbf{w}|\mathbf{v})$  into two parts,  $P(\mathbf{w}|\mathbf{a})$ , which we call the conditional language model given atoms, and  $P(\mathbf{a}|\mathbf{v})$ , which we call conditional atom model given visual inputs. To establish the oracle, we assume that the

<sup>2</sup><https://goo.gl/2hJ4lw>

atom model is given, which amounts to treat  $P(\mathbf{a}|\mathbf{v})$  as a Dirac delta function that assigns all the probability mass to the observed atom  $\mathbf{a}$ . In other words,  $P(\mathbf{w}|\mathbf{v}) = P_\theta(\mathbf{w}|\mathbf{a})$ .

Therefore, with the fully observed  $\mathbf{a}$ , the task of image and video captioning reduces to the task of language modeling conditioned on atoms. This is arguably a much easier task compared with the direct modeling of  $P(\mathbf{w}|\mathbf{v})$ , therefore a well-trained model could be treated as a performance oracle of it. Information contained in  $\mathbf{a}$  directly influences the difficulty of modeling  $P_\theta(\mathbf{w}|\mathbf{a})$ . For instance, if no atoms are available,  $P_\theta(\mathbf{w}|\mathbf{a})$  reduces to unconditional language modeling, which could be considered as a lower bound of  $P(\mathbf{w}|\mathbf{v})$ . By increasing the amount of information  $\mathbf{a}$  carries, the modeling of  $P_\theta(\mathbf{w}|\mathbf{a})$  becomes more and more straightforward.

### 7.3.1. Oracle Parameterization

Given a set of atoms  $\mathbf{a}^{(k)}$  that summarize the visual concept appearing in the visual inputs  $\mathbf{v}$ , this section describes the detailed parameterization of the model  $P_\theta(\mathbf{w}|\mathbf{a}^{(k)})$  with  $\theta$  denoting the overall parameters. In particular, we adopt the commonly used encoder-decoder framework (Cho et al., 2014) to model this conditional based on the following simple factorization  $P_\theta(\mathbf{w}|\mathbf{a}^{(k)}) = \prod_{t=1}^T P_\theta(\mathbf{w}_t|\mathbf{w}_{<t}, \mathbf{a}^{(k)})$ .

Recurrent neural networks (RNNs) are natural choices when outputs are identified as sequences. We borrow the recent success from a variant of RNNs called Long-short term memory networks (LSTMs) first introduced in Hochreiter and Schmidhuber (1997a), formulated as the following

$$\begin{bmatrix} p(\mathbf{w}_t | \mathbf{w}_{<t}, \mathbf{a}^{(k)}) \\ \mathbf{h}_t \\ \mathbf{c}_t \end{bmatrix} = \psi(\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{w}_{t-1}, \mathbf{a}^{(k)}), \quad (7.3.1)$$

where  $\mathbf{h}_t$  and  $\mathbf{c}_t$  represent the RNN state and memory of LSTMs at timestep  $t$  respectively. Combined with the atom representation, Eq. (7.3.1) is implemented as following

$$\begin{aligned} \mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{E}_w [\mathbf{w}_{t-1}] + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{A}_f \Phi(\mathbf{a}^{(k)}) + \mathbf{b}_f), \\ \mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{E}_w [\mathbf{w}_{t-1}] + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{A}_i \Phi(\mathbf{a}^{(k)}) + \mathbf{b}_i), \\ \mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{E}_w [\mathbf{w}_{t-1}] + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{A}_o \Phi(\mathbf{a}^{(k)}) + \mathbf{b}_o), \\ \tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c \mathbf{E}_w [\mathbf{w}_{t-1}] + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{A}_c \Phi(\mathbf{a}^{(k)}) + \mathbf{b}_c), \\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \\ \mathbf{h}_t &= \mathbf{o}_t \odot \mathbf{c}_t, \end{aligned}$$

where  $\mathbf{E}_w$  denotes the word embedding matrix, as apposed to the atom embedding matrix  $\mathbf{E}_a$ ,  $\mathbf{W}$ ,  $\mathbf{U}$ ,  $\mathbf{A}$  and  $\mathbf{b}$  are parameters of the LSTM. With the LSTM's state  $\mathbf{h}_t$ , the probability of the next word in the sequence is  $\mathbf{p}_t = \text{softmax}(\mathbf{U}_p \tanh(\mathbf{W}_p \mathbf{h}_t + \mathbf{b}_p) + \mathbf{d})$  with parameters



$\mathbf{U}_p$ ,  $\mathbf{W}_p$ ,  $\mathbf{b}_p$  and  $\mathbf{d}$ . The overall training criterion of the oracle is

$$\theta = \arg \max_{\theta} \mathcal{U}_k(\theta) = \log \prod_{n=1}^N P_{\theta}(\mathbf{w}^{(n)} | \mathbf{a}^{(n,k)}) = \sum_{n=1}^N \sum_{t=1}^T \log P_{\theta}(\mathbf{w}_t^{(n)} | \mathbf{w}_{<t}^{(n)}, \mathbf{a}^{(n,k)}), \quad (7.3.2)$$

given  $N$  training pairs  $(\mathbf{w}^{(n)}, \mathbf{a}^{(n,k)})$ .  $\theta$  represents parameters in the LSTM.

### 7.3.2. Atoms Construction

Each configuration of  $\mathbf{a}$  may be associated with a different distribution  $P_{\theta}(\mathbf{w} | \mathbf{a})$ , therefore a different oracle model. We define configuration as an orderless collection of unique atoms. That is,  $\mathbf{a}^{(k)} = \{a_1, \dots, a_k\}$  where  $k$  is the size of the bag and all items in the bag are different from each other. Considering the particular problem of image and video captioning, atoms are defined as words in captions that are most related to actions, entities, and attributes of entities (in Figure 7.1). The reason of using these three particular choices of language components as atoms is not an arbitrary decision. It is reasonable to consider these three types among the most visually perceivable ones when human describes visual content in natural language. We further verify this by conducting a human evaluation procedure to identify “visual” atoms from this set and show that a dominant majority of them indeed match human visual perception, detailed in Section 7.5.1. Being able to capture these important concepts is considered as crucial in getting superior performance. Therefore, comparing the performance of existing models against this oracle reveals their ability of capturing atoms from visual inputs when  $P(\mathbf{a} | \mathbf{v})$  is unknown.

A set of atoms  $\mathbf{a}^{(k)}$  is treated as “a bag of words”. As with the use of word embedding matrix in neural language modeling (Bengio et al., 2003), the  $i$ th atom  $\mathbf{a}_i^{(k)}$  is used to index the atom embedding matrix  $\mathbf{E}_a[\mathbf{a}_i^{(k)}]$  to obtain a vector representation of it. Then the representation of the entire set of atoms is  $\Phi(\mathbf{a}^{(k)}) = \sum_{i=1}^k \mathbf{E}_a[\mathbf{a}_i^{(k)}]$ .

## 7.4. CONTRIBUTING FACTORS OF THE ORACLE

The formulation of Section 7.3 is generic, only relying on the assumption the two-step visual captioning process, independent of the parameterization in Section 7.3.1. In practice, however, one needs to take into account several contributing factors to the oracle.

Firstly, atoms, or visual concepts, may be defined as 1-gram words, 2-gram phrases and so on. Arguably a mixture of N-gram representations has the potential to capture more complicated correlations among visual concepts. For simplicity, this work uses only 1-gram representations, detailed in Section 7.5.1. Secondly, the procedure used to extract atoms needs to be reliable, extracting mainly *visual* concepts, leaving out *non-visual* concepts. To ensure this, the procedure used in this work is verified with human evaluation, detailed in 7.5.1. Thirdly, the modeling capacity of the conditional language  $P_{\theta}(\mathbf{w} | \mathbf{a}^{(k)})$  has a direct influence on the obtained oracle. Section 7.3.1 has shown one example of many possible

parameterizations. Lastly, the oracle may be sensitive to the training procedure and its hyper-parameters (see Section 7.5.2).

While it is therefore important to keep in mind that proposed oracle *conditions* on the above factors, quite surprisingly, however, with the simplest procedure and parameterization we show in the experimental section that oracle serves their purpose reasonably well.

## 7.5. EXPERIMENTS

We demonstrate the procedure of learning the oracle on three standard visual captioning datasets. MS COCO (Lin et al., 2014) is the most commonly used benchmark dataset in image captioning. It consists of 82,783 training and 40,504 validation images. each image accompanied by 5 captions, all in one sentence. We follow the split used in Xu et al. (2015) where a subset of 5,000 images are used as validation, and another subset of 5,000 images are used for testing. YouTube2Text is the most commonly used benchmark dataset in video captioning. It consists of 1,970 video clips, each accompanied with multiple captions. Overall, there are 80,000 video and caption pairs. Following Yao et al. (2015b), it is split into 1,200 clips for training, 100 for validation and 670 for testing. Another two video captioning datasets have been recently introduced in Torabi et al. (2015) and Rohrbach et al. (2015a). Compared with YouTube2Text, they are both much larger in the number of video clips, most of which are associated with one or two captions. Recently they are merge together for Large Scale Movie Description Challenge (LSMDC). <sup>3</sup> We therefore name this particular dataset LSMDC. The official splits contain 91,908 clips for training, 6,542 for validation and 10,053 for testing.

### 7.5.1. Atom extraction

Visual concepts in images or videos are summarized as atoms that are provided to the caption language model. They are split into three categories: actions, entities, and attributes. To identify these three classes, we utilize Stanford natural language parser <sup>4</sup> to automatically extract them. After a caption is parsed, we apply simple heuristics based on the tags produced by the parser, ignoring the phrase and sentence level tags <sup>5</sup>: Use words tagged with {"NN", "NNP", "NNPS", "NNS", "PRP"} as entity atoms. Use words tagged with {"VB", "VBD", "VBG", "VBN", "VBP", "VBZ"} as action atoms. Use words tagged with {"JJ", "JJR", "JJS"} as attribute atoms. After atoms are identified, they are lemmatized with NLTK lemmatizer <sup>6</sup> to unify them to their original dictionary format <sup>7</sup>. Figure 7.1 illustrates some results. We extracted atoms for COCO, YouTube2Text and LSMDC. This gives 14,207

<sup>3</sup><https://goo.gl/2hJ4lw>

<sup>4</sup><http://goo.gl/lSvPr>

<sup>5</sup>complete list of tags: <https://goo.gl/fU8zDd>

<sup>6</sup><http://www.nltk.org/>

<sup>7</sup>available at <https://goo.gl/t7vtFj>



visual input	caption	entity	action	attribute
	A blue train sitting along side of a green forest.	train, side, forest	sit	blue, green
	Old train left out on the ground has graffiti all over it	train, it, ground	leave, graffiti, have	old
	Two abandoned blue and white train cars next to trees.	car, train, tree	abandon	blue, white, next
	A man with a skateboard under him, not touching, are in the air.	air, skateboard, him, man	touch	NA
	Two guys are skateboarding and performing jumps and tricks.	jump, trick, guy	perform, skateboard	NA
	Two men doing tricks on skateboards at the skate park	trick, park, men, skateboard	do	skate

FIG. 7.1. Given ground truth captions, three categories of visual atoms (entity, action and attribute) are automatically extracted using NLP Parser. “NA” denotes the empty atom set.

entities, 4,736 actions and 8,671 attributes for COCO, 6,922 entities, 2,561 actions, 2,637 attributes for YouTube2Text, and 12,895 entities, 4,258 actions, 8550 attributes for LSMDC. Note that although the total number of atoms of each categories may be large, atom frequency varies. In addition, the language parser does not guarantee the perfect tags. Therefore, when atoms are being used in training the oracle, we sort them according to their frequency and make sure to use more frequent ones first to also give priority to atoms with larger coverage, detailed in Section 7.5.2 below.

We conducted a simple human evaluation <sup>8</sup> to confirm that extracted atoms are indeed predominantly visual. As it might be impractical to evaluate all the extracted atoms for all three datasets, we focus on top 150 frequent atoms. This evaluation intends to match the last column of Table 7. II where current state-of-the-art models have the equivalent capacity of capturing perfectly less than 100 atoms from each of three categories. Subjects are asked to cast their vote independently. The final decision of an atom being visual or not is made by majority vote. Table 7. I shows the ratio of atoms flagged as visual by such procedure.

TAB. 7. I. Human evaluation of proportion of atoms that are voted as visual. It is clear that extracted atoms from three categories contain dominant amount of visual elements, hence verifying the procedure described in Section 7.3.2. Another observation is that entities and actions tend to be more visual than attributes according to human perception.

	entities	actions	attributes
COCO	92%	85%	81%
YouTube2Text	95%	91%	72%
LSMDC	83%	87%	78%

<sup>8</sup>details available at <https://goo.gl/t7vtFj>

### 7.5.2. Training

After the atoms are extracted, they are sorted according to the frequency they appear in the dataset, with the most frequent one leading the sorted list. Taking first  $k$  items from this list gives the top  $k$  most frequent ones, forming a bag of atoms denoted by  $\mathbf{a}^{(k)}$  where  $k$  is the size of the bag. Conditioned on the atom bag, the oracle is maximized as in Equ (7.3.2).

To form captions, we used a vocabulary of size 20k, 13k and 25k for COCO, YouTube2Text and LSMDC respectively. For all three datasets, models were trained on training set with different configuration of (1) atom embedding size, (2) word embedding size and (3) LSTM state and memory size. To avoid overfitting we also experimented weight decay and Dropout (Hinton et al., 2012b) to regularize the models with different size. In particular, we experimented with random hyper-parameter search by Bergstra and Bengio (2012) with range [128, 1024] on (1), (2) and (3). Similarly we performed random search on the weight decay coefficient with a range of  $[10^{-6}, 10^{-2}]$ , and whether or not to use dropout. Optimization was performed by SGD, minibatch size 128, and with Adadelta (Zeiler, 2012) to automatically adjust the per-parameter learning rate. Model selection was done on the standard validation set, with an early stopping patience of 2,000 (early stop if no improvement made after 2,000 minibatch updates). We report the results on the test splits.

### 7.5.3. Interpretation

All three metrics – BLEU, METEOR and CIDER are computed with Microsoft COCO Evaluation Server (Chen et al., 2015). Figure 7.2 summarizes the learned oracle with an increasing number of  $k$ .

#### 7.5.3.1. *comparing oracle performance with existing models*

We compare the current state-of-the-art models’ performance against the established oracles in Figure 7.2. Table 7. II shows the comparison on three different datasets. With Figure 7.2, one could easily associate a particular performance with the equivalent number of atoms perfectly captured across all 3 atom categories, as illustrated in Table 7. II, the oracle included in bold. It is somehow surprising that state-of-the-art models have performance that is equivalent to capturing only a small amount of “ENT” and “ALL”. This experiment highlights the shortcoming of the state-of-art visual models. By improving them, we could close the performance gap that we currently have with the oracles.

#### 7.5.3.2. *quantify the diminishing return*

As the number of atoms  $k$  in  $\mathbf{a}^{(k)}$  increases, one would expect the oracle to be improved accordingly. It is however not yet clear the speed of such improvement. In other words, the

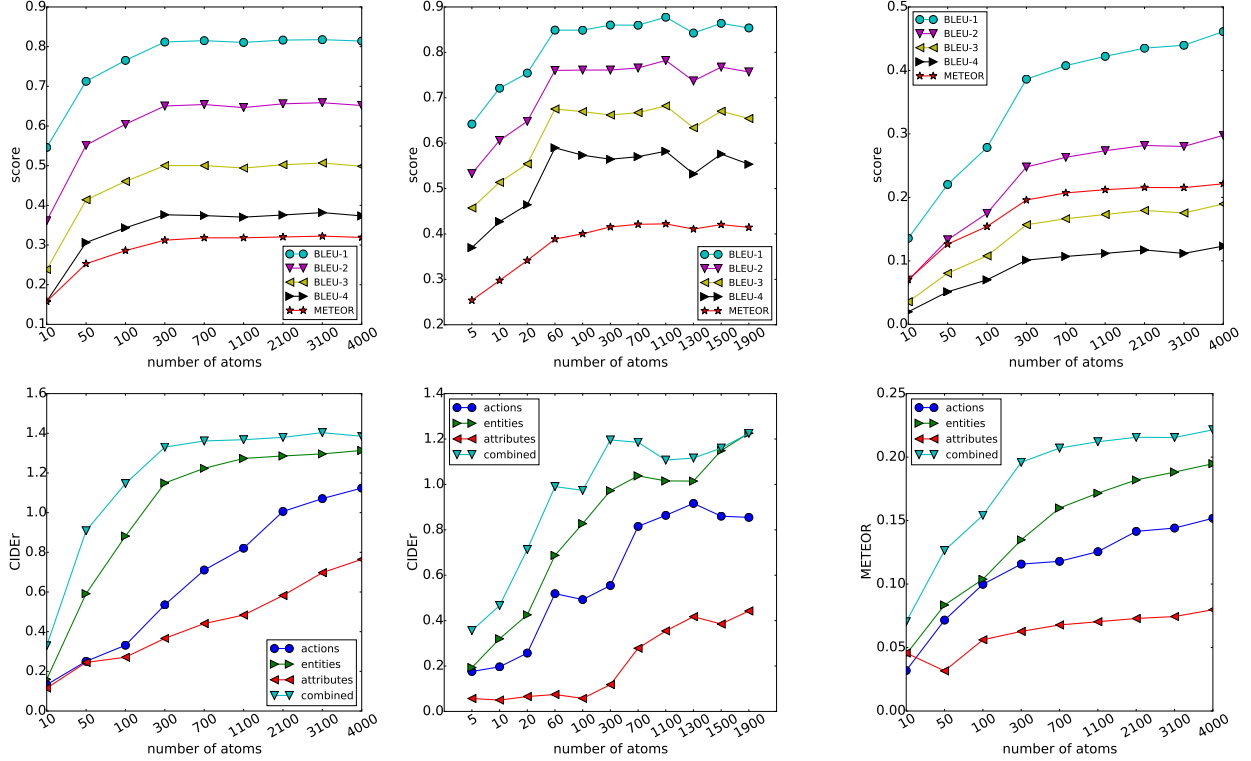


FIG. 7.2. Learned oracle on COCO (left), YouTube2Text (middle) and LSMDC (right). The number of atoms  $a^{(k)}$  is varied on x-axis and oracles are computed on y-axis on testsets. The first row shows the oracles on BLEU and METEOR with  $3k$  atoms,  $k$  from each of the three categories. The second row shows the oracles when  $k$  atoms are selected individually for each category. CIDEr is used for COCO and YouTube2Text as each test example is associated with multiple ground truth captions, argued in (Vedantam et al., 2014). For LSMDC, METEOR is used, as argued by Rohrbach et al. (2015).

gain in performance may not be proportional to the number of atoms given when generating captions, due to atom frequencies and language modeling. Figure 7.2 quantifies this effect. The oracle on all three datasets shows a significant gain at the beginning and diminishes quickly as more and more atoms are used.

Row 2 of Figure 7.2 also highlights the difference among actions, entities and attributes in generating captions. For all three datasets tested, entities played much more important roles, even more so than action atoms in general. This is particularly true on LSMDC where the gain of modeling attributes is much less than the other two categories.

Although visual atoms dominant the three atom categories shown in Section 7.5.1, as they increase in number, more and more non-visual atoms may be included, such as “living”, “try”, “find” and “free” which are relatively difficult to be associated with a particular part of visual inputs. Excluding non-visual atoms in the conditional language model can further tighten the oracle bound as less hints are provided to it. The major difficulty lies in the labor

TAB. 7. II.

Measure semantic capacity of current state-of-the-art models. One could easily map the reported metric to the number of visual atoms captured. This establishes an equivalence between a model, the proposed oracle and a model’s semantic capacity. (“ENT” for entities. “ACT” for actions. “ATT” for attributes. “ALL” for all three categories combined. “B1” for BLEU-1, “B4” for BLEU-4. “M” for METEOR. “C” for CIDEr. Note that the CIDEr is between 0 and 10. The learned oracle is denoted in **bold**.

	B1	B4	M	C	ENT	ACT	ATT	ALL
COCO	0.74 <b>0.80</b>	0.31 <b>0.35</b>	0.26 <b>0.30</b>	0.94 <b>1.4</b>	~200	~2100	>4000	~ 50
YouTube2Text	0.815 <b>0.88</b>	0.499 <b>0.58</b>	0.326 <b>0.40</b>	0.658 <b>1.2</b>	~60	~500	>1900	~ 20
LSMDC	N/A <b>0.45</b>	N/A <b>0.12</b>	0.07 <b>0.22</b>	N/A <b>N/A</b>	~40	~50	~4000	~10

of hand-separating visual atoms from non-visual ones as to the our best knowledge this is difficult to automate with heuristics.

#### 7.5.3.3. *atom accuracy versus atom quantity*

We have assumed that the atoms are given, or in other words, the prediction accuracy of atoms is 100%. In reality, one would hardly expect to have a perfect atom classifier. There is naturally a trade-off between number of atoms one would like to capture and the prediction accuracy of it. Figure 7.3 quantifies this trade-off on COCO and LSMDC. It also indicates the upper limit of performance given different level of atom prediction accuracy. In particular, we have replaced  $\mathbf{a}^{(k)}$  by  $\hat{\mathbf{a}}_r^{(k)}$  where  $r$  portion of  $\mathbf{a}^{(k)}$  are randomly selected and replaced by other randomly picked atoms not appearing in  $\mathbf{a}^{(k)}$ . The case of  $r = 0$  corresponds to those shown in Figure 7.2. And the larger the ratio  $r$ , the worse the assumed atom prediction is. The value of  $r$  is shown in the legend of Figure 7.3. According to the figure, in order to improve the caption generation score, one would have two options, either by keeping the number of atoms fixed while improving the atom prediction accuracy or by keeping the accuracy while increasing the number of included atoms. As state-of-art visual model already model around 1000 atoms, we hyphotesize that we could gain more in improving the atoms accuracy rather than increase the number of atom detected by those models.

#### 7.5.3.4. *intrinsic difficulties of particular datasets*

Figure 7.2 also reveals the intrinsic properties of each dataset. In general, the bounds on YouTube2Text are much higher than COCO, with LSMDC the lowest. For instance, from the first column of the figure, taking 10 atoms respectively, BLUE-4 is around 0.15 for COCO, 0.30 for YouTube2Text and less than 0.05 for LSMDC. With little visual information

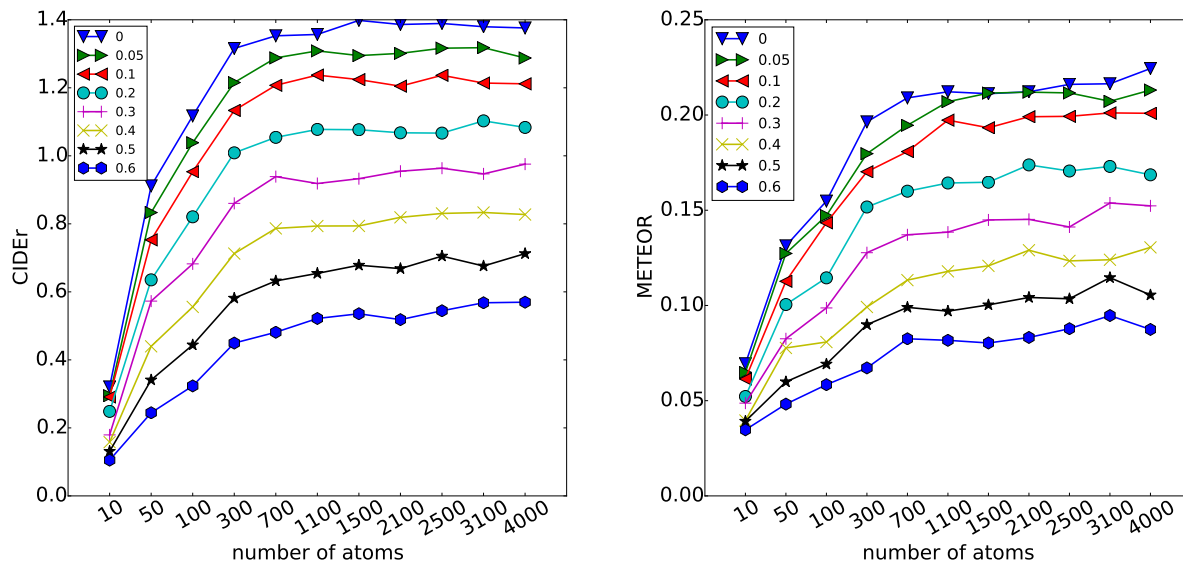


FIG. 7.3. Learned oracles with different atom precision ( $r$  in red) and atom quantity (x-axis) on COCO (left) and LSMDC (right). The number of atoms  $\hat{\mathbf{a}}_r^{(k)}$  is varied on x-axis and oracles are computed on y-axis on testsets. CIDEr is used for COCO and METEOR for LSMDC. It shows one could increase the score by either improving  $P(\mathbf{a}^{(k)}|\mathbf{v})$  with a fixed  $k$  or increase  $k$ . It also shows the maximal error bearable for different score.

to condition upon, a strong language model is required, which makes a dramatic difference across three datasets. Therefore the oracle, when compared across different datasets, offer an objective measure of difficulties of using them in the captioning task.

## 7.6. DISCUSSION

This work formulates oracle performance for visual captioning. The oracle is constructed with the assumption of decomposing visual captioning into two consecutive steps. We have assumed the perfection of the first step where visual atoms are recognized, followed by the second step where language models conditioned on visual atoms are trained to maximize the probability of given captions. Such an empirical construction requires only automatic atom parsing and the training of conditional language models, without extra labeling or costly human evaluation.

Such an oracle enables us to gain insight on several important factors accounting for both success and failure of the current state-of-the-art models. It further reveals model independent properties on different datasets. We furthermore relax the assumption of prefect atom prediction. This sheds light on a trade-off between atom accuracy and atom coverage, providing guidance to future research in this direction. Importantly, our experimental results suggest that more efforts are required in step one where visual inputs are converted into visual concepts (atoms).



Despite its effectiveness shown in the experiments, the empirical oracle is constructed with the simplest atom extraction procedure and model parameterization in mind, which makes such a construction in a sense a “conservative” oracle.

## ACKNOWLEDGMENTS

The authors would like to acknowledge the support of the following agencies for research funding and computing support: IBM T.J. Watson Research, NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs and CIFAR. We would also like to thank the developers of Theano ([Theano Development Team, 2016](#)) , for developing such a powerful tool for scientific computing.



# Chapter 8

---

## PROLOGUE TO THIRD ARTICLE

### 8.1. ARTICLE DETAIL

**Delving deeper into convolutional networks for learning video representations**  
Ballas, Nicolas and Yao, Li and Pal, Chris and Courville, Aaron, *International Conference on Learning Representations (ICLR)*, 2016.

*Personal contributions.* In the early days, Dr. Ballas and I experimented the idea of using a convolutional LSTM to generate low-resolution movies and were able to obtain decent amount of success (better than results reported in [Srivastava et al. \(2015a\)](#) with the bouncing MNIST, unpublished). This leads to his idea of applying a convolutional LSTM to learn a better video representation. We designed two sets of experiments to demonstrate the effectiveness of the proposed model. I coded, ran and wrote one of them – the part on video captioning, outperforming the state-of-the-art models at the time of this publication. The remaining part of the paper was mainly written by Dr. Ballas while Prof. Courville and I contributed in the polish.

### 8.2. CONTEXT

In Chapter 5, a standard LSTM is used as a decoder to generate languages. Chapter 7 further proves that a strong language decoder indeed plays a critical role in the standard encoder-decoder pipeline, perhaps so strong that it overshadows the ineffectiveness of a video encoder.

In seeking more effective video encoders, this work proposed an approach to learn spatio-temporal features in videos from intermediate visual representations we call “percepts” using Gated-Recurrent-Unit Recurrent Networks (GRUs). Our method relies on percepts that are extracted from all levels of a deep convolutional network trained on the large ImageNet dataset. While high-level percepts contain highly discriminative information, they tend to have a low-spatial resolution. Low-level percepts, on the other hand, preserve a higher spatial resolution from which we can model finer motion patterns.

Using low-level percepts, however, can lead to high-dimensionality video representations. To mitigate this effect and control the number of parameters, we introduce a variant of the GRU model that leverages the convolution operations to enforce sparse connectivity of the model units and share parameters across the input spatial locations.

We empirically validate our approach on both Human Action Recognition and Video Captioning tasks. In particular, we achieve results equivalent to state-of-art on the YouTube2Text dataset using a simpler caption-decoder model and without extra 3D CNN features.

### 8.3. CONTRIBUTIONS

This is the very first paper that trains a convolutional LSTM on videos. Previous work (see the references in Section 9.4) focused solely on applying a standard LSTM with fully connected layer on either raw pixels across frames or higher-level representations of them. The convolutional prior makes sense in videos as action patterns tend to appear locally in both spatial and temporal dimensions. In addition, as is true for all convolutional models, the number of parameters is greatly reduced. The in-depth analysis of its computational complexity is available in Section 9.3.1.

## Chapter 9

---

# DELVING DEEPER INTO CONVOLUTIONAL NETWORKS FOR LEARNING VIDEO REPRESENTATIONS

### 9.1. INTRODUCTION

Video analysis and understanding represents a major challenge for computer vision and machine learning research. While previous work has traditionally relied on hand-crafted and task-specific representations (Wang et al., 2011; Sadanand and Corso, 2012), there is a growing interest in designing general video representations that could help solve tasks in video understanding such as human action recognition, video retrieval or video captioning (Tran et al., 2014).

Two-dimensional Convolutional Neural Networks (CNN) have exhibited state-of-art performance in still image tasks such as classification or detection (Simonyan and Zisserman, 2014c). However, such models discard temporal information that has been shown to provide important cues in videos (Wang et al., 2011). On the other hand, recurrent neural networks (RNN) have demonstrated the ability to understand temporal sequences in various learning tasks such as speech recognition (Graves and Jaitly, 2014) or machine translation (Bahdanau et al., 2014). Consequently, Recurrent Convolution Networks (RCN) (Srivastava et al., 2015b; Donahue et al., 2014; Ng et al., 2015) that leverage both recurrence and convolution have recently been introduced for learning video representation. Such approaches typically extract “visual percepts” by applying a 2D CNN on the video frames and then feed the CNN activations to an RNN in order to characterize the video temporal variation.

Previous works on RCNs has tended to focus on high-level visual percepts extracted from the 2D CNN top-layers. CNNs, however, hierarchically build-up spatial invariance through pooling layers (LeCun et al., 1998; Simonyan and Zisserman, 2014c) as Figure 9.2 highlights. While CNNs tends to discard local information in their top layers, frame-to-frame temporal variation is known to be smooth. The motion of video patches tend to be restricted to a

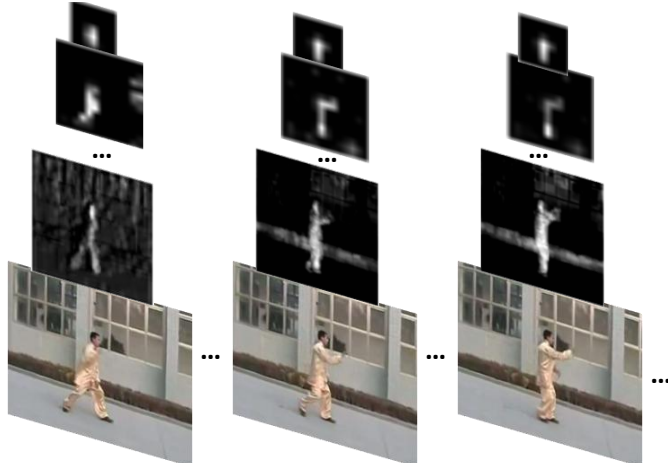


FIG. 9.1. Visualization of convolutional maps on successive frames in video. As we go up in the CNN hierarchy, we observe that the convolutional maps are more stable over time, and thus discard variation over short temporal windows.

local neighborhood (Brox and Malik, 2011). For this reason, we argue that current RCN architectures are not well suited for capturing fine motion information. Instead, they are more likely focus on global appearance changes such as shot transitions. To address this issue, we introduce a novel RCN architecture that applies an RNN not solely on the 2D CNN top-layer but also on the intermediate convolutional layers. Convolutional layer activations, or convolutional maps, preserve a finer spatial resolution of the input video from which local spatio-temporal patterns are extracted.

Applying an RNN directly on intermediate convolutional maps, however, inevitably results in a drastic number of parameters characterizing the input-to-hidden transformation due to the convolutional maps size. On the other hand, convolutional maps preserve the frame spatial topology. We propose to leverage this topology by introducing sparsity and locality in the RNN units to reduce the memory requirement. We extend the GRU-RNN model (Cho et al., 2014) and replace the fully-connected RNN linear product operation with a convolution. Our GRU-extension therefore encodes the locality and temporal smoothness prior of videos directly in the model structure.

We evaluate our solution on UCF101 human action recognition from Soomro et al. (2012) as well as the YouTube2text video captioning dataset from Chen and Dolan (2011b). Our experiments show that leveraging “percepts” at multiple resolutions to model temporal variation improves performance over our baseline model with respective gains of 3.4% for action recognition and 10% for video captioning.

## 9.2. GRU: GATED RECURRENT UNIT NETWORKS

In this section, we review Gated-Recurrent-Unit (GRU) networks which are a particular type of RNN. An RNN model is applied to a sequence of inputs, which can have variable lengths. It defines a recurrent hidden state whose activation at each time is dependent on that of the previous time. Specifically, given a sequence  $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T)$ , the RNN hidden state at time  $t$  is defined as  $\mathbf{h}_t = \phi(\mathbf{h}_{t-1}, \mathbf{x}_t)$ , where  $\phi$  is a nonlinear activation function. RNNs are known to be difficult to train due to the exploding or vanishing gradient effect (Bengio et al., 1994). However, variants of RNNs such as Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997a) or Gated Recurrent Units (GRU) (Cho et al., 2014) have empirically demonstrated their ability to model long-term temporal dependency in various task such as machine translation or image/video caption generation. In this paper, we will mainly focus on GRU networks as they have shown similar performance to LSTMs but with a lower memory requirement (Chung et al., 2014).

GRU networks allow each recurrent unit to adaptively capture dependencies of different time scales. The activation  $\mathbf{h}_t$  of the GRU is defined by the following equations:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1}), \quad (9.2.1)$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}), \quad (9.2.2)$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W} \mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1})), \quad (9.2.3)$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \mathbf{h}_{t-1} + \mathbf{z}_t \tilde{\mathbf{h}}_t, \quad (9.2.4)$$

where  $\odot$  is an element-wise multiplication.  $\mathbf{z}_t$  is an update gate that decides the degree to which the unit updates its activation, or content.  $\mathbf{r}_t$  is a reset gate.  $\sigma$  is the sigmoid function. When a unit  $r_t^i$  is close to 0, the reset gate forgets the previously computed state, and makes the unit act as if it is reading the first symbol of an input sequence.  $\tilde{\mathbf{h}}_t$  is a candidate activation which is computed similarly to that of the traditional recurrent unit in an RNN.

## 9.3. DELVING DEEPER INTO CONVOLUTIONAL NEURAL NETWORKS

This section delves into the main contributions of this work. We aim at leveraging visual percepts from different convolutional levels in order to capture temporal patterns that occur at different spatial resolution.

Let's consider  $(\mathbf{x}_t^1, \dots, \mathbf{x}_t^{L-1}, \mathbf{x}_t^L)_{(t=1..T)}$ , a set of 2D convolutional maps extracted from  $L$  layers at different time steps in a video. We propose two alternative RCN architectures, GRU-RCN, and Stacked-GRU-RCN (illustrated in Figure 9.2) that combines information extracted from those convolutional maps.

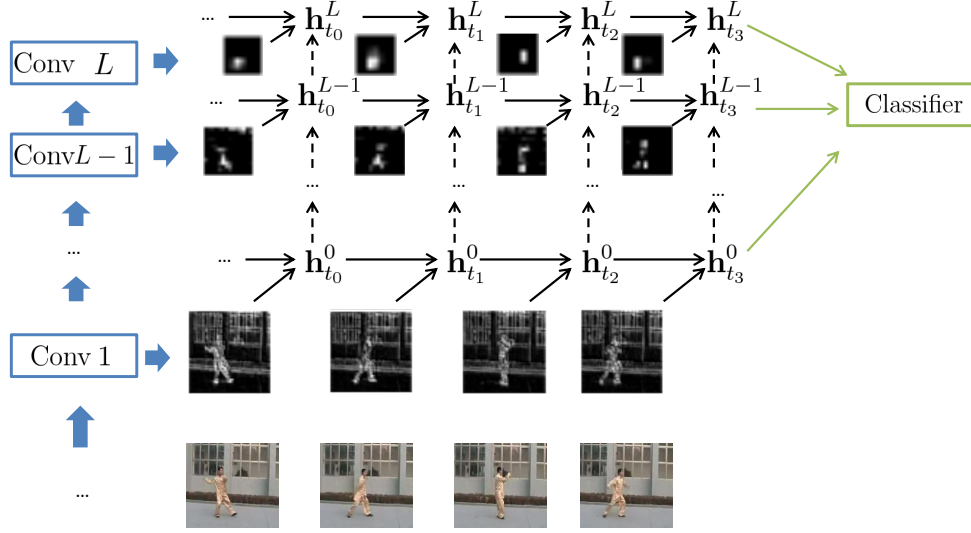


FIG. 9.2. High-level visualization of our model. Our approach leverages convolutional maps from different layers of a pretrained-convnet. Each map is given as input to a convolutional GRU-RNN (hence GRU-RCN) at different time-step. Bottom-up connections may be optionally added between RCN layers to form Stack-GRU-RCN.

### 9.3.1. GRU-RCN

In the first RCN architecture, we propose to apply  $L$  RNNs independently on each convolutional map. We define  $L$  RNNs as  $\phi^1, \dots, \phi^L$ , such that  $\mathbf{h}_t^l = \phi^l(\mathbf{x}_t^l, \mathbf{h}_{t-1}^l)$ . The hidden representation of the final time step  $\mathbf{h}_T^1, \dots, \mathbf{h}_T^L$  are then fed to a classification layer in the case of action recognition, or to a text-decoder RNN for caption generation.

To implement the RNN recurrent function  $\phi^l$ , we propose to leverage Gated Recurrent Units (Cho et al., 2014). GRUs were originally introduced for machine translation. They model input to hidden-state and hidden to hidden transitions using fully connected units. However, convolutional map inputs are 3D tensors (spatial dimension and input channels). Applying a GRU directly can lead to a drastic number of parameters. Let  $N_1$ ,  $N_2$  and  $O_x$  be the input convolutional map spatial size and number of channels. Applying a GRU directly would require input-to-hidden parameters  $\mathbf{W}^l$ ,  $\mathbf{W}_z^l$  and  $\mathbf{W}_r^l$  to be of size  $N_1 \times N_2 \times O_x \times O_h$  where  $O_h$  is the dimensionality of the GRU hidden representation.

Fully-connected GRUs do not take advantage of the underlying structure of convolutional maps. Indeed, convolutional maps are extracted from images that are composed of patterns with strong local correlation which are repeated over different spatial locations. In addition, videos have smooth temporal variation over time, *i.e.* motion associated with a given patch in successive frames will be restricted in a local spatial neighborhood. We embed such a prior in our model structure and replace the fully-connected units in GRU with convolution operations. We therefore obtain recurrent units that have sparse connectivity and share their

parameters across different input spatial locations:

$$\mathbf{z}_t^l = \sigma(\mathbf{W}_z^l * \mathbf{x}_t^l + \mathbf{U}_z^l * \mathbf{h}_{t-1}^l), \quad (9.3.1)$$

$$\mathbf{r}_t^l = \sigma(\mathbf{W}_r^l * \mathbf{x}_t^l + \mathbf{U}_r^l * \mathbf{h}_{t-1}^l), \quad (9.3.2)$$

$$\tilde{\mathbf{h}}_t^l = \tanh(\mathbf{W}^l * \mathbf{x}_t^l + \mathbf{U} * (\mathbf{r}_t^l \odot \mathbf{h}_{t-1}^l)), \quad (9.3.3)$$

$$\mathbf{h}_t^l = (1 - \mathbf{z}_t^l) \mathbf{h}_{t-1}^l + \mathbf{z}_t^l \tilde{\mathbf{h}}_t^l, \quad (9.3.4)$$

where  $*$  denotes a convolution operation. In this formulation, Model parameters  $\mathbf{W}$ ,  $\mathbf{W}_z^l$ ,  $\mathbf{W}_r^l$  and  $\mathbf{U}^l$ ,  $\mathbf{U}_z^l$ ,  $\mathbf{U}_r^l$  are 2D-convolutional kernels. Our model results in hidden recurrent representation that preserves the spatial topology,  $\mathbf{h}_t^l = (\mathbf{h}_t^l(i, j))$  where  $\mathbf{h}_t^l(i, j)$  is a feature vector defined at the location  $(i, j)$ . To ensure that the spatial size of the hidden representation remains fixed over time, we use zero-padding in the recurrent convolutions.

Using convolution, parameters  $\mathbf{W}^l$ ,  $\mathbf{W}_z^l$  and  $\mathbf{W}_r^l$  have a size of  $k_1 \times k_2 \times O_x \times O_h$  where  $k_1 \times k_2$  is the convolutional kernel spatial size (usually  $3 \times 3$ ), chosen to be significantly lower than convolutional map size  $N_1 \times N_2$ . The candidate hidden representation  $\tilde{\mathbf{h}}_t(i, j)$ , the activation gate  $\mathbf{z}_k(i, j)$  and the reset gate  $\mathbf{r}_k(i, j)$  are defined based on a local neighborhood of size  $(k_1 \times k_2)$  at the location  $(i, j)$  in both the input data  $\mathbf{x}_t$  and the previous hidden-state  $\mathbf{h}_{t-1}$ . In addition, the size of receptive field associated with  $\mathbf{h}^l(i, j)_t$  increases in the previous presentation  $\mathbf{h}_{t-1}^l, \mathbf{h}_{t-2}^l \dots$  as we go back further in time. Our model is therefore capable of characterizing spatio-temporal patterns with high spatial variation in time.

A GRU-RCN layer applies 6 2D-convolutions at each time-step (2 per GRU gate and 2 for computing the candidate activation). If we assume for simplicity that the input-to-hidden and hidden-to-hidden convolutions have the same kernel size and preserve the input dimension, GRU-RCN requires  $O(3TN_1N_2k_1k_2(O_xO_h + O_hO_h))$  multiplications. GRU-RCN sparse connectivity therefore saves computation compared to a fully-connected RNN that would require  $O(3TN_1N_2N_1N_2(O_xO_h + O_hO_h))$  computations. Memorywise, GRU-RCN needs to store the parameters for all 6 convolutions kernels leading to  $O(3k_1k_2(O_xO_h + O_hO_h))$  parameters.

### 9.3.2. Stacked GRU-RCN:

In the second RCN architecture, we investigate the importance of bottom-up connection across RNNs. While GRU-RCN applies each layer-wise GRU-RNN in an independent fashion, Stacked GRU-RCN preconditions each GRU-RNN on the output of the previous GRU-RNN at the current time step:  $\mathbf{h}_t^l = \phi^l(\mathbf{h}_{t-1}^l, \mathbf{h}_t^{l-1}, \mathbf{x}_t^l)$ . The previous RNN hidden representation is given as an extra-input to the GRU convolutional units:

$$\mathbf{z}_t^l = \sigma(\mathbf{W}_z^l * \mathbf{x}_t^l + \mathbf{W}_{z^l}^l * \mathbf{h}_t^{l-1} + \mathbf{U}_z^l * \mathbf{h}_{t-1}^l), \quad (9.3.5)$$

$$\mathbf{r}_t^l = \sigma(\mathbf{W}_r^l * \mathbf{x}_t^l + \mathbf{W}_{r^l}^l * \mathbf{h}_t^{l-1} + \mathbf{U}_r^l * \mathbf{h}_{t-1}^l), \quad (9.3.6)$$

$$\tilde{\mathbf{h}}_t^l = \tanh(\mathbf{W}^l * \mathbf{x}_t^l + \mathbf{U}^l * (\mathbf{r}_t \odot \mathbf{h}_{t-1}^l)), \quad (9.3.7)$$

$$\mathbf{h}_t^l = (1 - \mathbf{z}_t^l) \mathbf{h}_{t-1}^l + \mathbf{z}_t^l \tilde{\mathbf{h}}_t^l, \quad (9.3.8)$$

Adding this extra-connection brings more flexibility and gives the opportunity for the model to leverage representations with different resolutions.

## 9.4. RELATED WORK

Deep learning approaches have recently been used to learn video representations and have produced state-of-art results (Karpathy et al., 2014; Simonyan and Zisserman, 2014b; Wang et al., 2015; Tran et al., 2014). Karpathy et al. (2014); Tran et al. (2014) proposed to use 3D CNN learn a video representations, leveraging large training datasets such as the Sport 1 Million. However, unlike image classification (Simonyan and Zisserman, 2014c), CNNs did not yield large improvement over these traditional methods (Lan et al., 2014) highlighting the difficulty of learning video representations even with large training dataset. Simonyan and Zisserman (2014b) introduced a two-stream framework where they train CNNs independently on RGB and optical flow inputs. While the flow stream focuses only on motion information, the RGB stream can leverage 2D CNN pre-trained on image datasets. Based on the Two Stream representation, (Wang et al., 2015) extracted deep feature and conducted trajectory constrained pooling to aggregate convolutional feature as video representations.

RNN models have also been used to encode temporal information for learning video representations in conjunction with 2D CNNs. (Ng et al., 2015; Donahue et al., 2014) applied an RNN on top of the the two-stream framework, while (Srivastava et al., 2015b) proposed, in addition, to investigate the benefit of learning a video representation in an unsupervised manner. Previous works on this topic has tended to focus only on high-level CNN “visual percepts”. In contrast, our approach proposes to leverage visual “percepts” extracted from different layers in the 2D-CNN.

Recently, (Shi et al., 2015) also proposed to leverage convolutional units inside an RNN network. However, they focus on different task (now-casting) and a different RNN model based on an LSTM. In addition, they applied their model directly on pixels. Here, we use recurrent convolutional units on pre-trained CNN convolutional maps, to extract temporal pattern from visual “percepts” with different spatial sizes.

## 9.5. EXPERIMENTATION

This section presents an empirical evaluation of the proposed GRU-RCN and Stacked GRU-RCN architectures. We conduct experimentations on two different tasks: human action recognition and video caption generation.



### 9.5.1. Action Recognition

We evaluate our approach on the UCF101 dataset (Soomro et al., 2012). This dataset has 101 action classes spanning over 13320 YouTube videos clips. Videos composing the dataset are subject to large camera motion, viewpoint change and cluttered backgrounds. We report results on the dataset UCF101 first split, as this is most commonly used split in the literature. To perform proper hyperparameter search, we use the videos from the UCF-Thumos validation split (Jiang et al., 2014) as the validation set.

#### 9.5.1.1. Model Architecture

In this experiment, we consider the RGB and flow representations of videos as inputs. We extract visual “percept” using VGG-16 CNNs that consider either RGB or flow inputs. VGG-16 CNNs are pretrained on ImageNet (Simonyan and Zisserman, 2014c) and fine-tuned on the UCF-101 dataset, following the protocol in Wang et al. (2015). We then extract the convolution maps from *pool2*, *pool3*, *pool4*, *pool5* layers and the fully-connected map from layer *fc-7* (which can be view as a feature map with a  $1 \times 1$  spatial dimension). Those features maps are given as inputs to our RCN models.

We design and evaluate three RCN architectures for action recognition. In the first RCN architecture, GRU-RCN, we apply 5 convolutional GRU-RNNs independently on each convolutional map. Each convolution in the GRU-RCN has zero-padded  $3 \times 3$  convolutions that preserves the spatial dimension of the inputs . The number of channels of each respective GRU-RNN hidden-representations are 64, 128, 256, 256, 512. After the RCN operation we obtain 5 hidden-representations for each time step. We apply average pooling on the hidden-representations of the last time-step to reduce their spatial dimension to  $1 \times 1$ , and feed the representations to 5 classifiers, composed by a linear layer with a softmax nonlineary. Each classifier therefore focuses on only 1 hidden-representation extracted from the convolutional map of a specific layer. The classifier outputs are then averaged to get the final decision. A dropout ratio of 0.7 is applied on the input of each classifiers.

In the second RCN architecture, Stacked GRU-RCN, we investigate the usefulness of bottom-up connections. Our stacked GRU-RCN uses the same base architecture as the GRU-RCN, consisting of 5 convolutional GRU-RNNs having 64, 128, 256, 256 channels respectively. However, each convolutional GRU-RNN is now preconditioned on the hidden-representation that the GRU-RNN applied on the previous convolution-map outputs. We apply max-pooling on the hidden representations between the GRU-RNN layers for the compatibility of the spatial dimensions. As for the previous architecture, each GRU-RNN hidden-representation at the last time step is pooled and then given as input to a classifier.

Finally, in our bi-directional GRU-RCN, we investigate the importance of reverse temporal information. Given convolutional maps extracted from one layer, we run the GRU-RCN

twice, considering the inputs in both sequential and reverse temporal order. We then concatenate the last hidden-representations of the forward GRU-RCN and backward GRU-RCN, and give the resulting vector to a classifier.

#### 9.5.1.2. Model Training and Evaluation

We follow the training procedure introduced by the two-stream framework (Simonyan and Zisserman, 2014b). At each iteration, a batch of 64 videos are sampled randomly from the training set. To perform scale-augmentation, we randomly sample the cropping width and height from 256, 224, 192, 168. The temporal cropping size is set to 10. We then resize the cropped volume to  $224 \times 224 \times 10$ . We estimate each model parameters by maximizing the model log-likelihood:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N \log p(y^n | c(\mathbf{x}^n), \boldsymbol{\theta}),$$

where there are  $N$  training video-action pairs  $(\mathbf{x}^n, y^n)$ ,  $c$  is a function that takes a crop at random. We use Adam (Kingma and Ba, 2014) with the gradient computed by the backpropagation algorithm. We perform early stopping and choose the parameters that maximize the log-probability of the validation set.

We also follow the evaluation protocol of the two-stream framework (Simonyan and Zisserman, 2014b). At the test time, we sample 25 equally spaced video sub-volumes with a temporal size of 10 frames. From each of these selected sub-volumes, we obtain 10 inputs for our model, i.e. 4 corners, 1 center, and their horizontal flipping. The final prediction score is obtained by averaging across the sampled sub-volumes and their cropped regions.

#### 9.5.1.3. Results

We compare our approach with two different baselines, VGG-16 and VGG-16 RNN. VGG-16 is the 2D spatial stream that is described in (Wang et al., 2015). We take the VGG-16 model, pretrained on Image-Net and fine-tune it on the UCF-101 dataset. VGG-16 RNN baseline applied an RNN, using fully-connected gated-recurrent units, on top-of VGG-16. It takes as input the VGG-16 fully-connected representation  $fc-7$ . Following GRU-RCN top-layer, the VGG-16 RNN has hidden-representation dimensionality of 512.

The first column of Table 9.1 focuses on RGB inputs. We first report results of different GRU-RCN variants and compare them with the two baselines: VGG-16 and VGG-16 RNN. Our GRU-RCN variants all outperform the baselines, showing the benefit of delving deeper into a CNN in order to learn a video representation. We notice that VGG-16 RNN only slightly improve over the VGG-16 baseline, 78.1 against 78.0. This result confirms that CNN top-layer tends to discard temporal variation over short temporal windows. Stacked-GRU RCN performs significantly lower than GRU-RCN and Bi-directional GRU-RCN. We

Method	RGB	Flow
VGG-16	78.0	85.4
VGG-16 RNN	78.1	84.9
GRU-RCN	79.9	<b>85.7</b>
Stacked-GRU RCN	78.3	-
Bi-directional GRU-RCN	<b>80.7</b>	-
Two-Stream (Simonyan and Zisserman, 2014c)	72.8	81.2
Two-Stream + LSTM (Donahue et al., 2014)	71.1	76.9
Two-Stream + LSTM + Unsupervised (Srivastava et al., 2015b)	77.7	83.7
Improved Two-Stream (Wang et al., 2015)	<b>79.8</b>	<b>85.7</b>
C3D one network (Tran et al., 2014), 1 million videos as training	82.3	-
C3D ensemble (Tran et al., 2014), 1 million videos as training	<b>85.2</b>	-
Deep networks (Karpathy et al., 2014), 1 million videos as training	65.2	-

TAB. 9. I. Classification accuracy of different variants of the model on the UCF101 split 1. We report the performance of previous works that learn representation using only RGB information only.

argue that bottom-up connection, increasing the depth of the model, combined with the lack of training data (UCF-101 is train set composed by only 9500 videos) make the Stacked-GRU RCN learning difficult. The bi-directional GRU-RCN performs the best among the GRU-RCN variant with an accuracy of 80.7, showing the advantage of modeling temporal information in both sequential and reverse order. Bi-directional GRU-RCN obtains a gain 3.4% in term of performances, relatively to the baselines that focus only the VGG-16 top layer.

Table 9. I also reports results from other state-of-art approaches using RGB inputs. C3D (Tran et al., 2014) obtains the best performance on UCF-101 with 85.2. However, it should be noted that C3D is trained over 1 million videos. Other approaches use only the 9500 videos of UCF101 training set for learning temporal pattern. Our Bi-directional GRU-RCN compare favorably with other Recurrent Convolution Network (second blocks), confirming the benefit of using different CNN layers to model temporal variation.

Table 9. I also evaluates the GRU-RCN model applied flow inputs. VGG-16 RNN baseline actually decreases the performance compared to the VGG-16 baseline. On the other hand, GRU-RCN outperforms the VGG-16 baseline achieving 85.7 against 85.4. While the improvement is less important than the RGB stream, it should be noted that the flow stream of VGG-16 is applied on 10 consecutive flow inputs to extract visual “percepts”, and therefore already captures some motion information.

Finally, we investigate the combination of the RGB and flow streams. Following (Wang et al., 2015), we use a weighted linear combination of their prediction scores, where the weight is set to 2 as for the flow stream net and 1 for the temporal stream. Fusion the VGG-16 model baseline achieve an accuracy of 89.1. Combining the RGB Bi-directional GRU-RCN

with the flow GRU-RCN achieves a performance gain of 1.9% over baseline, reaching 90.8. Our model is on par with (Wang et al., 2015) that obtain state-of-art results using both RGB and flow streams which obtains 90.9.

### 9.5.2. Video Captioning

We also evaluate our representation on the video captioning task using YouTube2Text video corpus (Chen and Dolan, 2011b). The dataset has 1,970 video clips with multiple natural language descriptions for each video clip. The dataset is open-domain and covers a wide range of topics such as sports, animals, music and movie clips. Following (Yao et al., 2015a), we split the dataset into a training set of 1,200 video clips, a validation set of 100 clips and a test set consisting of the remaining clips.

#### 9.5.2.1. Model Specifications

To perform video captioning, we use the so-called encoder-decoder framework (Cho et al., 2014). In this framework the encoder maps input videos into abstract representations that precondition a caption-generating decoder.

As for encoder, we compare both VGG-16 CNN and Bi-directional GRU-RCN. Both models have been fine-tuned on the UCF-101 dataset and therefore focus on detecting *actions*. To extract an abstract representation from a video, we sample  $K$  equally-space segments. When using the VGG-16 encoder, we provide the *fc7* layer activations of the each segment’s first frame as the input to the text-decoder. For the GRU-RCN, we apply our model on the segment’s 10 first frames. We concatenate the GRU-RCN hidden-representation from the last time step. The concatenated vector is given as the input to the text decoder. As it has been shown that characterizing *entities* in addition of *action* is important for the caption-generation task (Yao et al., 2015), we also use as encoder a CNN (Szegedy et al., 2014), pretrained on ImageNet, that focuses on detecting static visual object categories.

As for the decoder, we use an LSTM text-generator with soft-attention on the video temporal frames (Yao et al., 2015a).

#### 9.5.2.2. Training

For all video captioning models, we estimated the parameters  $\theta$  of the decoder by maximizing the log-likelihood:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^N \sum_{i=1}^{t_n} \log p(y_i^n | y_{<i}^n, \mathbf{x}^n, \theta),$$

where there are  $N$  training video-description pairs  $(\mathbf{x}^n, y^n)$ , and each description  $y^n$  is  $t_n$  words long. We used Adadelta (Zeiler, 2012) We optimized the hyperparameters (e.g. number of LSTM units and the word embedding dimensionality, number of segment  $K$ ) using

	YouTube2Text			
Model	model selection	BLEU	METEOR	CIDEr
VGG-16 Encoder	BLEU	0.3700	0.2640	0.4330
Bi-directional GRU-RCN Encoder	BLEU	0.4100	0.2850	0.5010
GoogleNet Encoder	BLEU	0.4128	0.2900	0.4804
GoogleNet + Bi-directional GRU-RCN Encoder	BLEU	<b>0.4963</b>	0.3075	0.5937
GoogleNet + Bi-directional GRU-RCN Encoder	NLL	0.4790	0.3114	<b>0.6782</b>
GoogleNet + Bi-directional GRU-RCN Encoder	METEOR	0.4842	<b>0.3170</b>	0.6538
GoogleNet + Bi-directional GRU-RCN Encoder	CIDEr	0.4326	<b>0.3160</b>	<b>0.6801</b>
GoogleNet + HRNE (Pan et al., 2015)	-	0.436	<b>0.321</b>	-
VGG + p-RNN (Yu et al., 2015)	-	0.443	0.311	-
VGG + C3D + p-RNN (Yu et al., 2015)	-	<b>0.499</b>	<b>0.326</b>	-
Soft-attention (Yao et al., 2015a)	-	0.4192	0.2960	0.5167
Venugopalan <i>et al.</i> (Venugopalan et al., 2015)	-	0.3119	0.2687	-
+ Extra Data (Flickr30k, COCO)	-	0.3329	0.2907	-
Thomason <i>et al.</i> (Thomason et al., 2014)	-	0.1368	0.2390	-

TAB. 9. II. Performance of different variants of the model on YouTube2Text for video captioning. Representations obtained with the proposed RCN architecture combined with decoders from Yao et al. (2015a) offer a significant performance boost, reaching the performance of the other state-of-the-art models.

random search (Bergstra and Bengio, 2012) to maximize the log-probability of the validation set.

#### 9.5.2.3. Results

Table 9. II reports the performance of our proposed method using three automatic evaluation metrics. These are BLEU in Papineni et al. (2002), METEOR in Denkowski and Lavie (2014) and CIDEr in Vedantam et al. (2014). We use the evaluation script prepared and introduced in Chen et al. (2015). All models are early-stopped based on the negative-log-likelihood (NLL) of the validation set. We then select the model that performs best on the validation set according to the metric at consideration.

The first two lines of Table 9. II compare the performances of the VGG-16 and Bi-directional GRU-RCN encoder. Results clearly show the superiority of the Bi-Directional GRU-RCN Encoder as it outperforms the VGG-16 Encoder on all three metrics. In particular, GRU-RCN Encoder obtains a performance gain of 10% compared to the VGG-16 Encoder according to the BLEU metric. Combining our GRU-RCN Encoder that focuses on *action* with a GoogleNet Encoder that captures visual *entities* further improve the performances.

Our GoogleNet + Bi-directional GRU-RCN approach significantly outperforms Soft-attention (Yao et al., 2015a) that relies on a GoogLeNet and cuboids-based 3D-CNN Encoder,

in conjunction to a similar soft-attention decoder. This result indicates that our approach is able to offer more effective representations. According to the BLEU metric, we also outperform other approaches using more complex decoder schemes such as spatial and temporal attention decoder (Yu et al., 2015) or a hierarchical RNN decoder (Pan et al., 2015). Our approach is on par with (Yu et al., 2015), without the need of using a C3D-encoder that requires training on large-scale video dataset.

## 9.6. CONCLUSION

In this work, we address the challenging problem of learning discriminative and abstract representations from videos. We identify and underscore the importance of modeling temporal variation from “visual percepts” at different spatial resolutions. While high-level percepts contain highly discriminative information, they tend to have a low-spatial resolution. Low-level percepts, on the other hand, preserve a higher spatial resolution from which we can model finer motion patterns. We introduce a novel recurrent convolutional network architecture that leverages convolutional maps, from all levels of a deep convolutional network trained on the ImageNet dataset, to take advantage of “percepts” from different spatial resolutions.

We have empirically validated our approach on the Human Action Recognition and Video Captioning tasks using the UCF-101 and YouTube2Text datasets. Experiments demonstrate that leveraging “percepts” at multiple resolutions to model temporal variation improve over our baseline model, with respective gain of 3.4% and 10% for the action recognition and video captions tasks using RGB inputs. In particular, we achieve results comparable to state-of-art on YouTube2Text using a simpler text-decoder model and without extra 3D CNN features.

## ACKNOWLEDGMENTS

The authors would like to acknowledge the support of the following agencies for research funding and computing support: NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs and CIFAR. We would also like to thank the developers of Theano (Bergstra et al., 2010; Bastien et al., 2012), for developing such a powerful tool for scientific computing.

# Chapter 10

---

## PROLOGUE TO FOURTH ARTICLE

### 10.1. ARTICLE DETAIL

**On the equivalence between deep nade and generative stochastic networks**, Yao, Li and Ozair, Sherjil and Cho, Kyunghyun and Bengio, Yoshua, *European Conference on Machine Learning (ECML)*, 2014.

**Personal contributions.** The idea of making a theoretical connection between GSNs and NADEs was originally suggested by Prof. Bengio. The experiments are coded and conducted by Sherjil and me, with me taking the lead role. The paper is written by Dr. Cho, Prof. Bengio and me.

### 10.2. CONTEXT

Neural Autoregressive Distribution Estimators (NADEs) have recently been shown as successful alternatives for modeling high dimensional multimodal distributions. One issue associated with NADEs is that they rely on a particular order of factorization for  $P(\mathbf{x})$ . This issue has been recently addressed by a variant of NADE called Orderless NADEs and its deeper version, Deep Orderless NADE. Orderless NADEs are trained based on a criterion that stochastically maximizes  $P(\mathbf{x})$  with all possible orders of factorizations. Unfortunately, ancestral sampling from deep NADE is very expensive, corresponding to running through a neural net separately predicting each of the visible variables given some others. This work makes a connection between this criterion and the training criterion for Generative Stochastic Networks (GSNs). It shows that training NADEs in this way also trains a GSN, which defines a Markov chain associated with the NADE model. Based on this connection, we show an alternative way to sample from a trained Orderless NADE that allows to trade-off computing time and quality of the samples: a 3 to 10-fold speedup (taking into account the waste due to correlations between consecutive samples of the chain) can be obtained without noticeably reducing the quality of the samples. This is achieved using a novel sampling procedure for GSNs called annealed GSN sampling, similar to tempering methods that combines fast

mixing (obtained thanks to steps at high noise levels) with accurate samples (obtained thanks to steps at low noise levels).

### 10.3. CONTRIBUTIONS

This paper introduced a new view of the orderless NADE training procedure as a GSN training procedure, which yields several interesting conclusions:

- The orderless NADE training procedure also trains a GSN model, where the transition operator randomly selects a subset of input variables to be resampled given the others.
- Whereas orderless NADE models really represent an ensemble of conditionals that are not all compatible, the GSN interpretation provides a coherent interpretation of the estimated distribution through the stationary distribution of the associated Markov chain.
- Whereas ancestral sampling in NADE is exact, it is very expensive for deep NADE models, multiplying computing cost (of running once through the neural network to make a prediction) by the number of visible variables. On the other hand, each step of the associated GSN Markov chain only costs running once through the predictor, but because each prediction is made in parallel for all the resampled variables, each such step is also less accurate, unless very few variables are resampled. This introduces a trade-off between accuracy and computation time that can be controlled. This was validated experimentally.
- A novel sampling procedure for GSNs was introduced, called annealed GSN sampling, which permits a better trade-off by combining high-noise steps with a sequence of gradually lower noise steps, as shown experimentally.



# Chapter 11

---

## ON THE EQUIVALENCE BETWEEN DEEP NADE AND GENERATIVE STOCHASTIC NETWORK

### 11.1. INTRODUCTION

Unsupervised representation learning and deep learning have progressed rapidly in recent years (Bengio et al., 2013). On one hand, supervised deep learning algorithms have achieved great success. The authors of (Krizhevsky et al., 2012a), for instance, claimed the state-of-the-art recognition performance in a challenging object recognition task using a deep convolutional neural network. Despite the promise given by supervised deep learning, its unsupervised counterpart is still facing several challenges (Bengio, 2013). A large proportion of popular unsupervised deep learning models are based on either directed or undirected graphical models with latent variables (Hinton and Salakhutdinov, 2006a; Hinton et al., 2006b; Salakhutdinov and Hinton, 2009b). One problem of these unsupervised models is that it is often intractable to compute the likelihood of a model exactly.

The Neural Autoregressive Distribution Estimator (NADE) was proposed in (Larochelle and Murray, 2011a) to avoid this problem of computational intractability. It was inspired by the early work in (Bengio and Bengio, 2000), which like NADE modeled a binary distribution by decomposing it into a product of multiple conditional distributions of which each is implemented by a neural network, with parameters, representations and computations shared across all these networks. These kinds of models therefore implement a fully connected directed graphical model, in which ancestral sampling of the joint distribution is simple (but not necessarily efficient when the number of variables, e.g., pixel images, is large). Consequently, unlike many other latent variable models, it is possible with such directed graphical models to compute the exact probability of an observation tractably. NADEs have since been extended to model distributions of continuous variables in (Uria et al., 2013b), called a real-valued NADE (RNADE) which replaces a Bernoulli distribution with a mixture of Gaussian distributions for each conditional probability (see ,e.g., (Bishop, 1994)). The authors of (Uria et al., 2013a) proposes yet another variant of NADE, called a Deep NADE,

that uses a *deep* neural network to compute the conditional probability of each variable. In order to make learning tractable, they proposed a modified training procedure that effectively trains an ensemble of multiple NADEs.

Another thread of unsupervised deep learning is based on the family of autoencoders (see, e.g., (Vincent et al., 2010)). The autoencoder has recently begun to be understood as a density estimator (Alain and Bengio, 2013; Bengio et al., 2013b). These works suggest that an autoencoder trained with some arbitrary noise in the input is able to learn the distribution of either continuous or discrete random variables. This perspective on autoencoders has been further extended to a generative stochastic network (GSN) proposed in (Bengio et al., 2014).

Unlike a more conventional approach of directly estimating the probability distribution of data, a GSN aims to learn a transition probability of a Markov Chain Monte Carlo (MCMC) sampler whose stationary distribution estimates the data generating distribution. The authors of (Bengio et al., 2014) were able to show that it is possible to learn the distribution of data with a GSN having a network structure inspired by a deep Boltzmann machine (DBM) (Salakhutdinov and Hinton, 2009c) using this approach. Furthermore, a recently proposed multi-prediction DBM (MP-DBM) (Goodfellow et al., 2013), which models the joint distribution of data instance and its label, can be considered a special case of a GSN and achieves state-of-the-art classification performance on several datasets.

In this paper, we find a close relationship between the deep NADE and the GSN. We show that training a deep NADE with the order-agnostic (OA) training procedure (Uria et al., 2013a) can be cast as GSN training. This equivalence allows us to have an alternative theoretical explanation of the OA training procedure. Also, this allows an alternative sampling procedure for a deep NADE based on a MCMC method, rather than ancestral sampling.

In Sec. 11.2.1 and Sec. 11.3, we describe both NADE and GSN in detail. Based on these descriptions we establish the connection between the order-agnostic training procedure for NADE and the training criterion of GSN in Sec. 11.4 and propose a novel sampling algorithm for deep NADE. In Sec. 11.5, we introduce a novel sampling strategy for GSN called annealed GSN sampling, which is inspired by tempering methods and does a good trade-off between computing time and accuracy. We empirically investigate the effect of the proposed GSN sampling procedure for deep NADE models in Sec. 11.6.

## 11.2. DEEP NADE AND ORDER-AGNOSTIC TRAINING

In this section we describe the deep NADE and its training criterion, closely following (Uria et al., 2013a).

### 11.2.1. NADE

NADE (Larochelle and Murray, 2011a) models a joint distribution  $p(\mathbf{x})$  where  $\mathbf{x} \in \mathbb{R}^D$ .  $D$  is the dimensionality of  $\mathbf{x}$ . NADE factorizes  $p(\mathbf{x})$  into

$$p(\mathbf{x}) = \prod_{d=1}^D p(x_{o_d} | \mathbf{x}_{o_{<d}}) \quad (11.2.1)$$

where  $o$  is a predefined ordering of  $D$  indices.  $o_{<d}$  denotes the first  $d - 1$  indices of the ordering  $o$ .

The NADE then models each factor in Eq. (11.2.1) with a neural network having a single hidden layer  $H$ . That is,

$$p(x_{o_d} = 1 | \mathbf{x}_{o_{<d}}) = \sigma(\mathbf{V}_{:,o_d} \mathbf{h}_d + b_{o_d}),$$

where

$$\mathbf{h}_d = \phi(\mathbf{W}_{:,o_{<d}} + \mathbf{c}).$$

$\mathbf{V} \in \mathbb{R}^{H \times D}$ ,  $b \in \mathbb{R}^D$ ,  $\mathbf{W} \in \mathbb{R}^{H \times D}$  and  $\mathbf{c} \in \mathbb{R}^H$  are the output weights, the output biases, the input weights and the hidden biases, respectively.  $\sigma$  is a logistic sigmoid function, and  $\phi$  can be any nonlinear activation function.

To train such a model, one maximizes the log-likelihood function of the training set

$$\theta^* = \arg \max_{\theta} \mathcal{L}_o(\theta) = \arg \max_{\theta} \frac{1}{N} \sum_{n=1}^N \sum_{d=1}^D \log p(x_{o_d}^n | \mathbf{x}_{o_{<d}}^n, o), \quad (11.2.2)$$

where  $\theta$  denotes all the parameters of the model.

### 11.2.2. Deep NADE

One issue with the original formulation of the NADE is that the ordering of variables needs to be predefined and fixed. Potentially, this limits the inference capability of a trained model such that when the model is asked to infer the conditional probability which is not one of the factors in the predefined factorization (See Eq. (11.2.1)). For instance, a NADE trained with  $D$  visible variables with an ordering  $(1, 2, \dots, D)$ , one cannot easily infer  $x_2 || x_1, x_D$  except by expensive (and intractable) marginalization over all the other variables.

Another issue is that it is not possible to build a deeper architecture for NADE with the original formulation without losing a lot in efficiency. When there is only a single hidden layer with  $H$  units in the neural network modeling each conditional probability of a NADE, it is possible to share the parameters (the input weights and the hidden biases) to keep the computational complexity linear with respect to the number of parameters, i.e.,  $O(DH)$ . However, if there are more than one hidden layers, it is not possible to re-use computations in the same way. In this case, the computational complexity is  $O(DH + DH^2L)$  where  $L$  is the number of hidden layers. Notice the extra  $D$  factor in front, compared to the number

of parameters which is  $O(DH + H^2L)$ . This comes about because we cannot re-use the computations performed after the first hidden layer for predicting the  $i$ -th variable, when predicting the following ones. In the one-layer case, this sharing is possible because the hidden units weighted sums needed when predicting the  $i + 1$ -th variable are the same as the weighted sums needed when predicting the  $i$ -th variable, plus the scalar contributions  $w_{ki}$  associated with the  $k$ -th hidden unit and the extra input  $x_i$  that is now available when predicting  $x_{i+1}$  but was not available when predicting  $x_i$ .

To resolve those two issues, the authors of (Uria et al., 2013a) proposed the order-agnostic (OA) training procedure that trains a factorial number of NADEs with shared parameters. In this case, the following objective function is maximized, instead of  $\mathcal{L}_o$  in Eq. (11.2.2):

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}^n} \sum_{d=1}^D \mathbb{E}_{o_{<d}} \mathbb{E}_{o_d} \log p(x_{o_d}^n | \mathbf{x}_{o_{<d}}^n, \theta, o). \quad (11.2.3)$$

This objective function is, however, intractable, since it involves the factorial number of summations. Instead, in practice, when training, we use a stochastic approximation  $\hat{\mathcal{L}}$  by sampling an ordering  $o$ , the index of predicted variable  $d$  and a training sample  $\mathbf{x}^n$  at each time:

$$\hat{\mathcal{L}}(\theta) = \frac{D}{D-d+1} \sum_{i \notin o_{<d}} \log p(x_i^n | \mathbf{x}_{o_{<d}}^n, \theta, o). \quad (11.2.4)$$

Computing  $\hat{\mathcal{L}}$  is identical to a forward computation in a regular feedforward neural network except for two differences. Firstly, according to the sampled ordering  $o$ , the input variables of indices  $o_{>d}$  are set to 0, and the identity of the zeroed indices is provided as extra inputs (through a binary vector of length  $D$ ). Secondly, the conditional probabilities of only those variables of indices  $o_{>d}$  are used to compute the objective function  $\hat{\mathcal{L}}$ .

This order-agnostic procedure solves the previously raised issues of the original NADE. Since the model is optimized for all possible orderings, it does not suffer from being inefficient at inferring any conditional probability. Furthermore, the lack of predefined ordering makes it possible to use a single set of parameters for modeling all conditional distributions. Thus, the computational cost of training a deep NADE with the OA procedure is linear with respect to the number of parameters, regardless of the depth of each neural network.

From here on, we call a NADE trained with the OA procedure simply a deep NADE to distinguish it from a NADE trained with a usual training algorithm other than the OA procedure.

### 11.3. GENERATIVE STOCHASTIC NETWORKS

In (Bengio et al., 2013b, 2014) a new family of models called generative stochastic networks (GSN) was proposed, which tackles the problem of modeling a data distribution,  $p(\mathbf{x})$ , although without providing a tractable expression for it.

The underlying idea is to learn a transition operator of a Markov Chain Monte Carlo (MCMC) sampler that samples from the distribution  $p(\mathbf{x})$ , instead of learning the whole distribution directly. If we let  $p(\mathbf{x}' | \mathbf{x})$  be the transition operator, then we may rewrite it by introducing a latent variable  $\mathbf{h}$  into

$$p(\mathbf{x}' | \mathbf{x}) = \sum_{\mathbf{h}} p(\mathbf{x}' | \mathbf{h})p(\mathbf{h} | \mathbf{x}). \quad (11.3.1)$$

In other words, two separate conditional distributions  $p(\mathbf{x}' | \mathbf{h})$  and  $p(\mathbf{h} | \mathbf{x})$  jointly define the transition operator. In (Bengio et al., 2013b, 2014) it is argued that it is easier to learn these simpler conditional distributions because they have less modes (they only consider small changes from the previous state), meaning that the associated normalization constants can be estimated more easily (either by an approximate parametrization, e.g., a single or few component mixture, or by MCMC on a more powerful parametrization, which will have less variance if the number of modes is small).

A special form of GSN also found with denoising auto-encoders predefines  $p(\mathbf{h} | \mathbf{x})$  such that it does not require learning from data. Then, we only learn  $p(\mathbf{x}' | \mathbf{h})$ . This is the case in (Bengio et al., 2013b), where they proposed to use a user-defined corruption process, such as randomly masking out some variables with a fixed probability, for  $p(\mathbf{h} | \mathbf{x})$ . They, then, estimated  $p(\mathbf{x}' | \mathbf{h})$  as a denoising autoencoder  $f_\theta$ , parameterized with  $\theta$ , that reverses the corruption process  $p(\mathbf{h} | \mathbf{x})$  (Vincent et al., 2008b).

It was shown in (Bengio et al., 2013b) that if the denoising process  $f_\theta$  is a consistent estimator of  $p(\mathbf{x}' | \mathbf{h})$ , this leads to consistency of the Markov chain's stationary distribution as an estimator of the data generating distribution. This is under some conditions ensuring the irreducibility, ergodicity and aperiodicity of the Markov chain, i.e., that it mixes. In other words, training  $f_\theta$  to match  $p(\mathbf{x}' | \mathbf{h})$  is enough to learn implicitly the whole distribution  $p(\mathbf{x})$ , albeit indirectly, i.e., through the definition of a Markov chain transition operator. The result from (Bengio et al., 2014) further suggests that it is possible to also parameterize the corruption process  $p(\mathbf{h} | \mathbf{x})$  and learn both  $p(\mathbf{h} | \mathbf{x})$  and  $p(\mathbf{x}' | \mathbf{h})$  together.

From the qualitative observation on some of the learned transition operators of GSNs (see, e.g., (Bengio et al., 2014)), it is clear that the learned transition operator quickly finds a plausible mode in the whole distribution, even when the Markov chain was started from a random configuration of  $\mathbf{x}$ . This is because the GSN reconstruction criterion encourages the learner to quickly move from low probability configurations to high-probability ones, i.e., to burn-in quickly. This is in contrast to using a Gibbs sampler to generate samples from other generative models that explicitly model the whole distribution  $p(\mathbf{x})$ , which requires often many more *burn-in* steps before the Markov chain finds a plausible mode of the distribution.

## 11.4. EQUIVALENCE BETWEEN DEEP NADE AND GSN

Having described both deep NADE and GSN, we now establish the relationship, or even equivalence, between them. In particular, we show in this section that the order-agnostic (OA) training procedure for NADE is one special case of GSN learning.

We start from the stochastic approximation to the objective function of the OA training procedure for deep NADE in Eq. (11.2.4). We notice that the sampled ordering  $o$  in the objective function  $\hat{\mathcal{L}}$  can be replaced with another random variable  $\mathbf{m} \in \{0, 1\}^D$ , where  $D$  is the dimensionality of an observation  $x$ . The binary mask  $\mathbf{m}$  is constructed such that

$$m_i = \begin{cases} 1, & \text{if } i \in o_{<d} \\ 0, & \text{otherwise} \end{cases}$$

Then, we rewrite Eq. (11.2.4) by

$$\begin{aligned} \hat{\mathcal{L}}(\theta) &\propto \sum_{i=1}^D (1 - m_i) \log p(x_i^n \mid \mathbf{m} \odot \mathbf{x}^n, \theta, \mathbf{m}) \\ &= \sum_{i=1}^D \log (m_i + (1 - m_i) p(x_i^n \mid \mathbf{m} \odot \mathbf{x}^n, \theta, \mathbf{m})) \\ &= \sum_{i=1}^D \log (m_i + (1 - m_i) p(x_i^n \mid \mathbf{h}^{(n)}, \theta)) \end{aligned} \quad (11.4.1)$$

where  $m_i$  is the  $i$ -th element of the binary mask  $\mathbf{m}$ , and  $\odot$  is an element-wise multiplication. We introduced a new variable  $\mathbf{h} = [\mathbf{m}, \mathbf{m} \odot \mathbf{x}^n] \in \mathbb{R}^{2D}$  which is a concatenation of the corrupted copy (some variables masked out) of  $\mathbf{x}^n$  and the sampled mask  $\mathbf{m}$ .

It is now easy to see the connection between the objective function of the OA training procedure in Eq. (11.4.1) to a GSN training criterion using a user-defined (not learned) corruption process which we described in the earlier section.

In this case, the corruption process  $p(\mathbf{h} \mid \mathbf{x})$  (Eq. (11.3.1)) is

$$p(\mathbf{h} \mid \mathbf{x}) = p([\mathbf{m}, \mathbf{m} \odot \mathbf{x}] \mid \mathbf{x}) = \prod_{i=1}^D k \prod_{j=1}^D \delta_{m_j x_j}(h_{j+D}), \quad (11.4.2)$$

where  $k$  is a random number sampled uniformly between 0 and 1, and  $\delta_\mu(a)$  is a shifted Dirac delta function which is 1 only when  $a = \mu$  and 0 otherwise. This means that sampling is done by first generating an uniformly random binary mask  $\mathbf{m}$  and then taking  $\mathbf{m} \odot \mathbf{x}$  as the corrupted version of  $\mathbf{x}$ .

The conditional probability of  $\mathbf{x}'$  given  $\mathbf{h}$  is

$$p(\mathbf{x}' \mid \mathbf{h}) = \prod_{i=1}^D [m_i \delta_{x_i}(x'_i) + (1 - m_i) p(x'_i \mid r_i(\mathbf{x} \odot \mathbf{m} \mid \theta))], \quad (11.4.3)$$

where  $r_i$  is a parametric function (neural network) that models the conditional probability.

If we view the estimation of  $p(\mathbf{x}' | \mathbf{h})$  in Eq. (11.4.3) as a denoising autoencoder, one effectively ignores each variable  $x'_i$  with its mask  $m_i$  set to 1, since the sample of  $x'_i$  from Eq. (11.4.3) is always  $x_i$  due to  $\delta_{x_i}(x'_i)$ . A high-capacity auto-encoder could learn that when  $m_i = 1$ , it can just copy the  $i$ -th input to the  $i$ -th output. On the other hand, when  $m_i$  is 0, training this denoising autoencoder would maximize  $\log p(x'_i | r_i(\mathbf{x} \odot \mathbf{m} | \theta))$ , making it assign high probability to the original  $x_i$  given the non-missing inputs. Therefore, maximizing the logarithm of  $p(\mathbf{x}' | \mathbf{h})$  in Eq. (11.4.3) is equivalent to maximizing  $\hat{L}$  in Eqs. (11.4.1) and (11.2.4) up to a constant.

In essence, maximizing  $\hat{\mathcal{L}}$  in Eq. (11.2.4) is equivalent to training a GSN with the conditional distributions defined in Eqs. (11.4.2)–(11.4.3). Furthermore, the chain defined in this way is ergodic as every state  $\mathbf{x}$  has a non-zero probability at each step ( $\mathbf{x} \rightarrow \mathbf{x}'$ ), making this GSN chain a valid MCMC sampler.

#### 11.4.1. Alternative Sampling Method for NADE

Although the training procedure of the deep NADE introduced in (Uria et al., 2013a) is order-agnostic, sampling from the deep NADE is not.

The authors of (Uria et al., 2013a) proposed an ancestral sampling method for a deep NADE. Firstly, one randomly selects an ordering uniformly from all possible orderings. One generates a sample of each variable from its conditional distribution following the selected ordering. When  $D$ ,  $H$  and  $L$  are respectively the dimensionality of the observation variable, the number of hidden units in each hidden layer and the number of layers, the time complexity of sampling a single sample using this ancestral approach is  $O(DLH^2)$ .

We propose here an alternative sampling strategy based on our observation of the equivalence between the deep NADE and GSN. The new strategy is simply to alternating between sampling from  $p(\mathbf{h} | \mathbf{x})$  in Eq. (11.4.2) and  $p(\mathbf{x}' | \mathbf{h})$  in Eq. (11.4.3), which corresponds to performing Markov Chain Monte Carlo (MCMC) sampling on  $p(\mathbf{x})$ . The computational complexity of a single step ( $\mathbf{x} \rightarrow \mathbf{h} \rightarrow \mathbf{x}'$ ) in this case is  $O(DH + LH^2)$ .

Unlike the original ancestral strategy, the proposed approach does not generate an exact sample in a single step. Instead, one often needs to run the chain  $K$  steps until the exact, independent sample from the stationary distribution of the chain is collected, which we call *burn-in*. In other words, the new approach requires  $O(KDH + K LH^2)$  to collect a single sample in the worst case.<sup>1</sup>

If we assume that  $H$  is not too larger than  $D$  ( $H = O(D)$  or  $H = \Theta(D)$ ), which is an usual practice, the time complexity of the ancestral approach is  $O(D^3)$ , and that of the proposed GSN approach is  $O(KD^2)$ , where we further assume that  $L$  is a small constant. Effectively, if the MCMC method used in the latter strategy requires only a small, controllable number  $K$

<sup>1</sup> Since it is a usual practice to collect every  $t$ -th samples from the same chain, where  $t \ll K$ , we often do not need  $KN$  steps to collect  $N$  samples.



of steps to generate a single exact, independent sample such that  $K \ll D$ , the new approach is more efficient in collecting samples from a trained deep NADE. Importantly, as we have already mentioned earlier, a GSN has been shown to learn a transition operator of an MCMC method that requires only a small number of burn-in steps.

In the experiments, we investigate empirically whether this new sampling strategy is computationally more efficient than the original ancestral approach in a realistic setting.

#### 11.4.2. The GSN Chain Averages an Ensemble of Density Estimators

As discussed in (Uria et al., 2013a), maximizing  $\hat{\mathcal{L}}$  in Eq. (11.2.3) can be considered as training a factorial number of different NADEs with shared parameters. Each NADE differs from each other by the choice of the ordering of variables and may assign a different probability to the same observation.<sup>2</sup> Based on this observation, it is suggested in (Uria et al., 2013a) to use the average of the assigned probabilities by all these NADE, or a small randomly chosen subset of them, as the actual probability.

This interpretation of seeing the deep NADE as an ensemble of multiple NADEs and our earlier argument showing that the deep NADE training is special case of GSN training naturally leads to a question: *does the GSN Markov chain average an ensemble of density/distribution estimators?*

We claim that the answer is yes. From the equivalence we showed in this paper, it is clear that a GSN trained with a criterion such as the NADE criterion *learns* an ensemble of density/distribution estimators (in this case, masking noise with the reconstruction conditional distribution in Eq. (11.4.3)). Furthermore, when one samples from the associated GSN Markov chain, one is averaging the contributions associated with different orders. So, although each of these conditionals (predicting a subset given another subset) may not be consistent with a single joint distribution, the associated GSN Markov chain which combines them randomly does define a clear joint distribution: the stationary distribution of the Markov chain. Clearly, this stationary distribution is an ensemble average over all the possible orderings.

### 11.5. ANNEALED GSN SAMPLING

With the above proposal for GSN-style sampling of a Deep NADE model, one can view the average fraction  $p$  of input variables that are resampled at each step as a kind of noise level, or the probability of resampling any particular visible variable  $x_i$ . With uniform sampling of subsets, we obtain  $p = 0.5$ , but both higher and lower values are possible. When  $p = 1$ , all variables are resampled independently and the resulting samples are coming from the marginal distributions of each variable, which would be a very poor rendering of

<sup>2</sup> The fact that all ensembles share the exact same parameters makes it similar to the recently proposed technique of dropout (Hinton et al., 2012a).





FIG. 11.1. Independent samples generated by the ancestral sampling procedure from the deep NADE.

the Deep NADE distribution, but would mix very well. With  $p$  as small as possible (or more precisely, resampling only one randomly chosen variable given the others), we obtain a *Gibbs sampler* associated with the Deep NADE distribution, which we know has the same stationary distribution as Deep NADE itself. However, this would mix very slowly and would not bring any computational gain over ancestral sampling in the Deep NADE model (in fact it would be considerably worse because the correlation between consecutive samples would reduce the usefulness of the Markov chain samples, compared to ancestral sampling that provides i.i.d. samples). With intermediate values of  $p$ , we obtain a compromise between the fast computation and the quality of samples.

However, an even better trade-off can be reached by adopting a form of annealed sampling for GSNs, a general recipe for improving the compromise between accuracy of the sampling distribution and mixing for GSNs. For this purpose we talk about a generic noise level, although in this paper we refer to  $p$ , the probability of resampling any particular visible variable.

The idea is inspired by annealing and tempering methods that have been useful for undirected graphical models (Neal, 1994, 2001): *before sampling from the low-noise regime, we run the high-noise version of the transition operator and gradually reduce the noise level over a sequence of steps.* The steps taken at high noise allow to mix quickly while the steps taken at low noise allow to burn-in near high probability samples. Therefore we consider an approximation of the GSN transition operator which consists of the successive application of a sequence of instances of the operator associated with gradually reduced noise levels, ending at the target noise level. Conceptually, it is as if the overall Markov chain was composed, for each of its steps, by a short chain of steps with gradually decreasing noise levels. By making the annealing schedule have several steps at or near the target low noise level, and

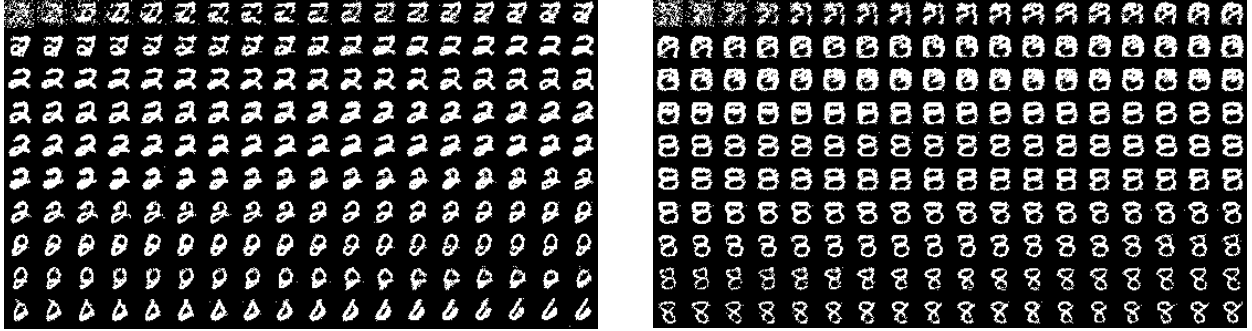


FIG. 11.2. The consecutive samples from the two independent GSN sampling chains without any annealing strategy. Both chains started from uniformly random configurations. Note the few spurious samples which can be avoided with the annealing strategy (see Figure 11.3).

by controlling the lengths of these annealing runs, we can trade-off between accuracy of the samples (improved by a longer annealing run length) and speed of computation.

In the experiments, we used the following annealing schedule:

$$p_t = \max(p_{\min}, p_{\max} - (t - 1) * (p_{\max} - p_{\min}) / (\alpha * (T - 1)))$$

where  $p_{\max}$  is the high noise level,  $p_{\min}$  is the low (target) noise level,  $T$  is the length of the annealing run, and  $\alpha \geq 1$  controls which fraction of the run is spent in annealing vs doing burn-in at the low noise level.

## 11.6. EXPERIMENTS

### 11.6.1. Settings: Dataset and Model

We run experiments using the handwritten digits dataset (MNIST) which has 60,000 training samples and 10,000 test samples. Each sample has 784 dimensions, and we binarized each variable by thresholding at 0.5. The training set is split into two so that the first set of 50,000 samples is used to train a model and the other set of 10,000 samples is used for validation.

Using MNIST we trained deep NADE with various architectures and sets of hyperparameters using the order-agnostic (OA) training procedure (see Sec. 11.2.2). The best deep NADE model according to the validation performance has two hidden layers with size 2000 and was trained with a linearly decaying learning rate schedule (from 0.001 to 0) for 1000 epochs. We use this model to evaluate the two sampling strategies described and proposed earlier in this paper.

### 11.6.2. Qualitative Analysis

Fig. 11.1 shows a subset of 10,000 samples collected from the deep NADE using the conventional ancestral sampling. The average log-probability of the samples is  $-70.36$  according to the deep NADE. As each sample by the ancestral sampling is exact and independent from others, we use these samples and their log-probability as a baseline for assessing the proposed GSN sampling procedure.

We first generate samples from the deep NADE using the GSN sampling procedure without any annealing strategy. A sampling chain is initialized with a uniformly random configuration, and a sample is collected at each step. The purpose of this sampling is to empirically confirm that the GSN sampling does not require many steps for burn-in. We ran two independent chains and visualize the initial 240 samples from each of them in Fig. 11.2, which clearly demonstrates that the chain rapidly finds a plausible mode in only a few steps.

Although this visualization suggests a faster burn-in, one weakness is clearly visible from these figures (Fig. 11.2). The chain generates many consecutive samples of a single digit before it starts generating samples of another digit. That is, the samples are highly correlated temporally, suggesting potentially slow convergence to the stationary distribution.

We then tried sampling from the deep NADE using the novel annealed GSN sampling proposed in Sec. 11.5. Fig. 11.3 visualizes the collected, samples over the *consecutive* annealing runs. Compared to the samples generated using the ordinary GSN sampling method, the chain clearly mixes well. One can hardly notice a case where a successive sample is a realization of the same digit from the previous sample. Furthermore, the samples are qualitatively comparable to those exact samples collected with the ancestral sampling (see Fig. 11.1).

In the following section, we further investigate the proposed annealed GSN sampling in a more quantitative way, in comparison to the ancestral sampling.

### 11.6.3. Quantitative Results

We first evaluate the effect of using a user-defined noise level in  $p(\mathbf{h} \mid \mathbf{x})$  (Eq. (11.4.2)). We generated 1000 samples from GSN chains with five different noise levels; 0.1, 0.3, 0.4, 0.5 and 0.6. For each noise level, we ran 100 independent chains and collected every 200-th sample from each chain. As a comparison, we also generated 1000 samples from a chain with the proposed annealed GSN sampling with  $p_{\max} = 0.9$ ,  $p_{\min} = 0.1$  and  $\alpha = 0.7$ .

We computed the log-probability of the set of samples collected from each chain with the deep NADE to evaluate the quality of the samples. Tab. 11. I lists the log-probabilities of the sets of samples, which clearly shows that as the noise level increases the quality of the samples degrades. Importantly, none of the chains were able to generate samples from the model that are close to those generated by the ancestral sampling. However, the annealed

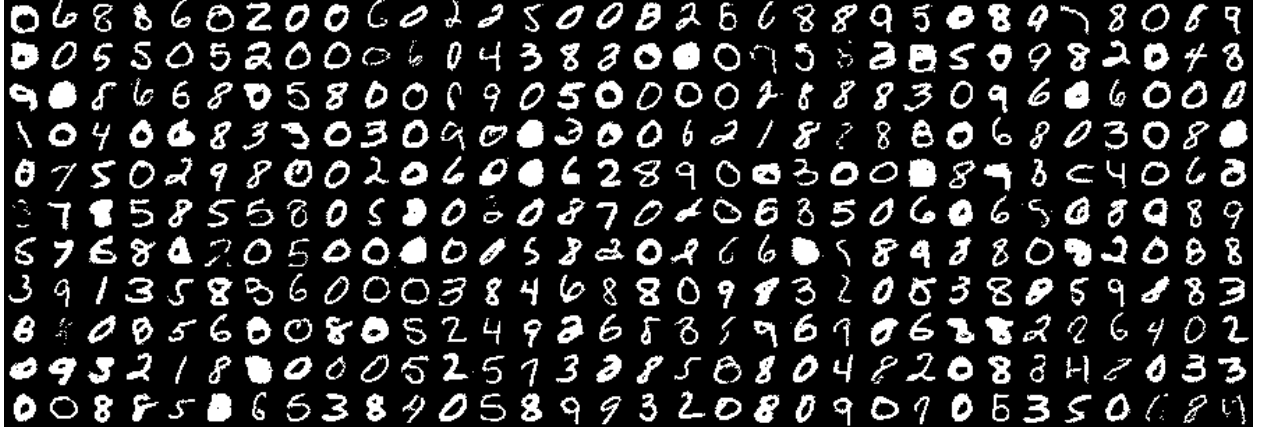


FIG. 11.3. Samples generated by the annealed GSN sampling procedure for the same deep NADE model. Visually the quality is comparable to the ancestral samples, and mixing is very fast. This is obtained with  $p_{\max} = 0.9$ ,  $p_{\min} = 0.1$ ,  $\alpha = 0.7$  and  $T = 20$ .

GSN sampling was able to generate samples that are quantitatively as good as those from the ancestral sampling.

Noise	Log-Probability
0.1	-77.1
0.3	-78.93
0.4	-77.9
0.5	-81.1
0.6	-88.1
Annealed	-69.72
Ancestral	-70.36

TAB. 11. I. Log-probability of 1000 samples when anealing is not used. To collect samples, 100 parallel chains are run and 10 samples are taken from each chain and combined together. The noise level is fixed at a particular level during the sampling. We also report the best log-probability of samples generated with an annealed GSN sampling.

We also perform quantitative analysis to measure the computational gain when using the GSN sampling procedure to generate samples. The speedup by using annealed GSN sampling instead of ancestral sampling is shown in Figure 11.4. To compute the speedup factor, we timed both the ancestral NADE sampling and GSN sampling on the same machine running single process. NADE sampling takes 3.32 seconds per sample and GSN sampling takes 0.009 seconds. That means the time to get one sample in ancestral sampling can get 369 samples in GSN sampling. Although the the direct speedup factor is 369, it must be discounted because of the autocorrelation of successive samples in the GSN chain. Then we perform different GSN sampling runs with different settings of  $\alpha$ . Figure 11.4 shows the results with different  $\alpha$ . For each  $\alpha$ , a GSN sampling starting at random is run and we collect one out of every  $K$  samples till 1000 samples are collected. The effective sample size (Geyer, 1992) is then estimated based on the sum of the autocorrelations in the autocorrelation factor. The speedup factor is discounted accordingly.

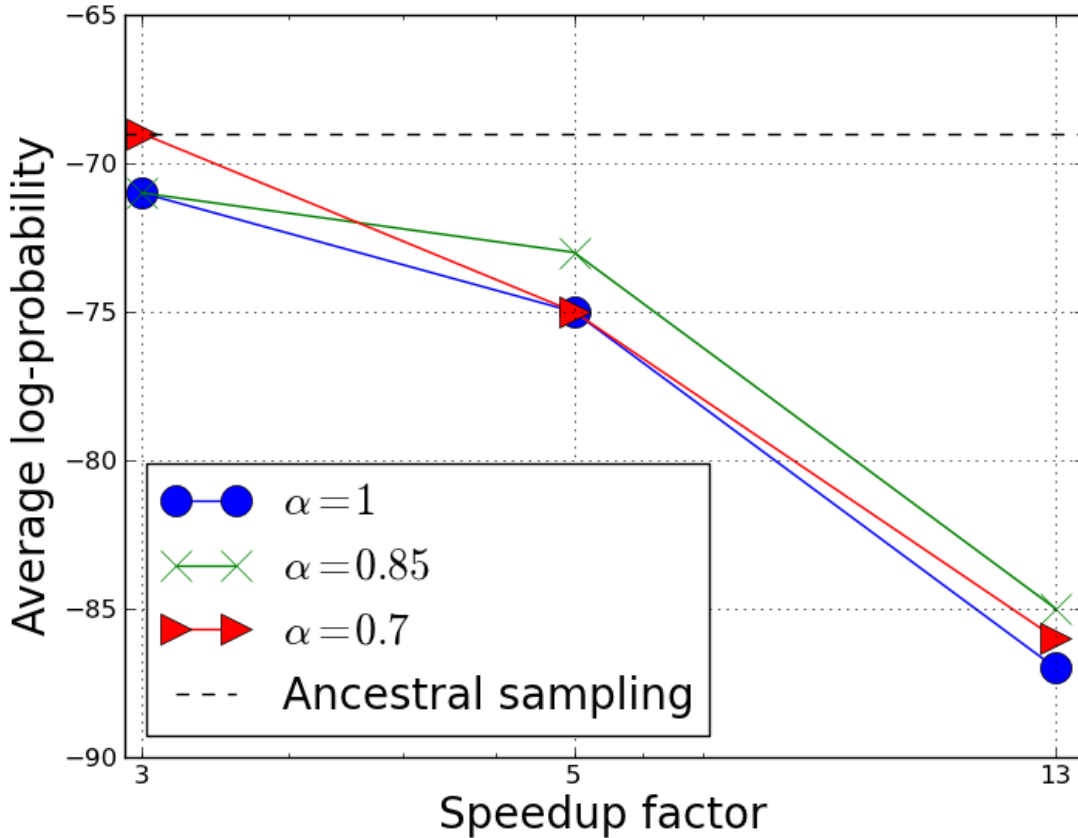


FIG. 11.4. The annealed GSN sampling procedure is compared against NADE ancestral sampling, trading off the computational cost (computational wrt to ancestral sampling on x-axis) against log-likelihood of the generated samples (y-axis). The computational cost discards the effect of the Markov chain autocorrelation by estimating the effective number of samples and increasing the computational cost accordingly.

## 11.7. CONCLUSIONS

This paper introduced a new view of the orderless NADE training procedure as a GSN training procedure, which yields several interesting conclusions:

- The orderless NADE training procedure also trains a GSN model, where the transition operator randomly selects a subset of input variables to be resampled given the others.
- Whereas orderless NADE models really represent an ensemble of conditionals that are not all compatible, the GSN interpretation provides a coherent interpretation of the estimated distribution through the stationary distribution of the associated Markov chain.
- Whereas ancestral sampling in NADE is exact, it is very expensive for deep NADE models, multiplying computing cost (of running once through the neural network to

make a prediction) by the number of visible variables. On the other hand, each step of the associated GSN Markov chain only costs running once through the predictor, but because each prediction is made in parallel for all the resampled variables, each such step is also less accurate, unless very few variables are resampled. This introduces a trade-off between accuracy and computation time that can be controlled. This was validated experimentally.

- A novel sampling procedure for GSNs was introduced, called annealed GSN sampling, which permits a better trade-off by combining high-noise steps with a sequence of gradually lower noise steps, as shown experimentally.

## ACKNOWLEDGMENTS

We would like to thank the developers of Theano ([Bergstra et al., 2010](#); [Bastien et al., 2012](#)). We would also like to thank CIFAR, and Canada Research Chairs for funding, and Compute Canada, and Calcul Québec for providing computational resources.

# Chapter 12

---

## PROLOGUE TO FIFTH ARTICLE

### 12.1. ARTICLE DETAIL

**Iterative neural autoregressive distribution estimator nade-k**, Raiko, Tapani and Li, Yao and Cho, Kyunghyun and Bengio, Yoshua, *Advances in neural information processing systems (NIPS)*, 2014.

**Personal contributions.** Prof. Raiko visited the lab and proposed the idea of viewing the NADE model as a special case of NADE-k where inference is performed with  $k$  steps of iterations. I implemented all the codes and performed all the necessary experiments in the paper. The early draft was written by me, and Prof. Raiko, Dr. Cho and Prof. Bengio helped to drastically improve the writing. As the original idea came from Prof. Raiko and he contributes the most in the final writing as well, he led the author list. It is however fair to state that my contribution is equally significant.

### 12.2. CONTEXT

In a broader context, this work explores the problem of unsupervised learning of image representations. Training of the neural autoregressive density estimator (NADE) can be viewed as doing one step of probabilistic inference on missing values in data. We propose a new model that extends this inference scheme to multiple steps, arguing that it is easier to learn to improve a reconstruction in  $k$  steps rather than to learn to reconstruct in a single inference step. The proposed model is an unsupervised building block for deep learning that combines the desirable properties of NADE and multi-prediction training: (1) Its test likelihood can be computed analytically, (2) it is easy to generate independent samples from it, and (3) it uses an inference engine that is a superset of variational inference for Boltzmann machines. The proposed NADE-k was competitive with the state-of-the-art in density estimation on the two datasets tested.

### 12.3. CONTRIBUTIONS

In this paper, we proposed a model called iterative neural autoregressive distribution estimator (NADE- $k$ ) that extends the conventional neural autoregressive distribution estimator (NADE) and its order-agnostic training procedure. The proposed NADE- $k$  maintains the tractability of the original NADE while we showed that it outperforms the original NADE as well as similar, but intractable generative models such as restricted Boltzmann machines and deep belief networks. Just like an iterative mean-field approximation in Boltzmann machines, the proposed NADE- $k$  performs multiple iterations through hidden layers and a visible layer to infer the probability of the missing value, unlike the original NADE which performs the inference of a missing value in a single iteration through hidden layers.

Our empirical results show that this approach of multiple iterations improves the performance of a model that has the same number of parameters, compared to performing a single iteration. This suggests that the inference method has significant effect on the efficiency of utilizing the model parameters. Also, we were able to observe that the generative performance of NADE can come close to more sophisticated models such as deep belief networks in our approach.



# Chapter 13

---

## ITERATIVE NEURAL AUTOREGRESSIVE DISTRIBUTION ESTIMATOR (NADE- $k$ )

### 13.1. INTRODUCTION

Traditional building blocks for deep learning have some unsatisfactory properties. Boltzmann machines are, for instance, difficult to train due to the intractability of computing the statistics of the model distribution, which leads to the potentially high-variance MCMC estimators during training (if there are many well-separated modes (Bengio et al., 2013)) and the computationally intractable objective function. Autoencoders have a simpler objective function (e.g., denoising reconstruction error (Vincent et al., 2010)), which can be used for model selection but not for the important choice of the corruption function. On the other hand, this paper follows up on the Neural Autoregressive Distribution Estimator (NADE, Larochelle and Murray, 2011b), which specializes previous neural auto-regressive density estimators (Bengio and Bengio, 2000) and was recently extended (Uria et al., 2014) to deeper architectures. It is appealing because both the training criterion (just log-likelihood) and its gradient can be computed tractably and used for model selection, and the model can be trained by stochastic gradient descent with backpropagation. However, it has been observed that the performance of NADE has still room for improvement.

The idea of using missing value imputation as a training criterion has appeared in three recent papers. This approach can be seen either as training an energy-based model to impute missing values well (Brakel et al., 2013), as training a generative probabilistic model to maximize a generalized pseudo-log-likelihood (Goodfellow et al., 2013), or as training a denoising autoencoder with a masking corruption function (Uria et al., 2014). Recent work on generative stochastic networks (GSNs), which include denoising auto-encoders as special cases, justifies dependency networks (Heckerman et al., 2000) as well as generalized pseudo-log-likelihood (Goodfellow et al., 2013), but have the disadvantage that sampling from the trained “stochastic fill-in” model requires a Markov chain (repeatedly resampling some subset of the values given the others). In all these cases, learning progresses by back-propagating

the imputation (reconstruction) error through inference steps of the model. This allows the model to better cope with a potentially imperfect inference algorithm. This learning-to-cope was introduced recently in 2011 by [Stoyanov et al. \(2011\)](#) and [Domke \(2011\)](#).

The NADE model involves an ordering over the components of the data vector. The core of the model is the reconstruction of the next component given all the previous ones. In this paper we reinterpret the reconstruction procedure as a single iteration in a variational inference algorithm, and we propose a version where we use  $k$  iterations instead, inspired by ([Goodfellow et al., 2013](#); [Brakel et al., 2013](#)). We evaluate the proposed model on two datasets and show that it outperforms the original NADE ([Larochelle and Murray, 2011b](#)) as well as NADE trained with the order-agnostic training algorithm ([Uria et al., 2014](#)).

### 13.2. PROPOSED METHOD: NADE- $k$

We propose a probabilistic model called NADE- $k$  for  $D$ -dimensional binary data vectors  $\mathbf{x}$ . We start by defining  $p_{\theta}$  for imputing missing values using a fully factorial conditional distribution:

$$p_{\theta}(\mathbf{x}_{\text{mis}} \mid \mathbf{x}_{\text{obs}}) = \prod_{i \in \text{mis}} p_{\theta}(x_i \mid \mathbf{x}_{\text{obs}}), \quad (13.2.1)$$

where the subscripts mis and obs denote missing and observed components of  $\mathbf{x}$ . From the conditional distribution  $p_{\theta}$  we compute the joint probability distribution over  $\mathbf{x}$  given an ordering  $o$  (a permutation of the integers from 1 to  $D$ ) by

$$p_{\theta}(\mathbf{x} \mid o) = \prod_{d=1}^D p_{\theta}(x_{o_d} \mid \mathbf{x}_{o_{<d}}), \quad (13.2.2)$$

where  $o_{<d}$  stands for indices  $o_1 \dots o_{d-1}$ .

The model is trained to minimize the negative log-likelihood averaged over all possible orderings  $o$

$$\mathcal{L}(\theta) = \mathbb{E}_{o \in D!} [\mathbb{E}_{\mathbf{x} \in \text{data}} [-\log p_{\theta}(\mathbf{x} \mid o)]] . \quad (13.2.3)$$

using an unbiased, stochastic estimator of  $\mathcal{L}(\theta)$

$$\hat{\mathcal{L}}(\theta) = -\frac{D}{D-d+1} \log p_{\theta}(\mathbf{x}_{o_{\geq d}} \mid \mathbf{x}_{o_{<d}}) \quad (13.2.4)$$

by drawing  $o$  uniformly from all  $D!$  possible orderings and  $d$  uniformly from  $1 \dots D$  ([Uria et al., 2014](#)). Note that while the model definition in Eq. (13.2.2) is sequential in nature, the training criterion (13.2.4) involves reconstruction of all the missing values in parallel. In this way, training does not involve picking or following specific orders of indices.

In this paper, we define the conditional model  $p_{\theta}(\mathbf{x}_{\text{mis}} \mid \mathbf{x}_{\text{obs}})$  using a deep feedforward neural network with  $nk$  layers, where we use  $n$  weight matrices  $k$  times. This can also be interpreted as running  $k$  successive inference steps with an  $n$ -layer neural network.

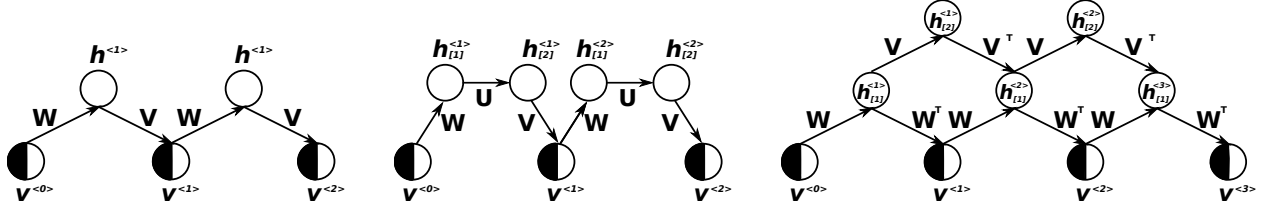


FIG. 13.1. The choice of a structure for NADE- $k$  is very flexible. The dark filled halves indicate that a part of the input is observed and fixed to the observed values during the iterations. Left: Basic structure corresponding to Equations (13.2.6–13.2.7) with  $n = 2$  and  $k = 2$ . Middle: Depth added as in NADE by Uria et al. (2014) with  $n = 3$  and  $k = 2$ . Right: Depth added as in Multi-Prediction Deep Boltzmann Machine by Goodfellow et al. (2013) with  $n = 2$  and  $k = 3$ . The first two structures are used in the experiments.

The input to the network is

$$\mathbf{v}^{(0)} = \mathbf{m} \odot \mathbb{E}_{\mathbf{x} \in \text{data}} [\mathbf{x}] + (\mathbf{1} - \mathbf{m}) \odot \mathbf{x} \quad (13.2.5)$$

where  $\mathbf{m}$  is a binary mask vector indicating missing components with 1, and  $\odot$  is an element-wise multiplication.  $\mathbb{E}_{\mathbf{x} \in \text{data}} [\mathbf{x}]$  is an empirical mean of the observations. For simplicity, we give equations for a simple structure with  $n = 2$ . See Fig. 13.1 (left) for the illustration of this simple structure.

In this case, the activations of the layers at the  $t$ -th step are

$$\mathbf{h}^{(t)} = \phi(\mathbf{W}\mathbf{v}^{(t-1)} + \mathbf{c}) \quad (13.2.6)$$

$$\mathbf{v}^{(t)} = \mathbf{m} \odot \text{sigmoid}(\mathbf{V}\mathbf{h}^{(t)} + \mathbf{b}) + (\mathbf{1} - \mathbf{m}) \odot \mathbf{x} \quad (13.2.7)$$

where  $\phi$  is an element-wise nonlinearity,  $\sigma$  is a logistic sigmoid function, and the iteration index  $t$  runs from 1 to  $k$ . The conditional probabilities of the variables (see Eq. (13.2.1)) are read from the output  $\mathbf{v}^{(k)}$  as

$$p_{\theta}(x_i = 1 \mid \mathbf{x}_{\text{obs}}) = v_i^{(k)}. \quad (13.2.8)$$

Fig. 13.2 shows examples of how  $\mathbf{v}^{(t)}$  evolves over iterations, with the trained model.

The parameters  $\theta = \{\mathbf{W}, \mathbf{V}, \mathbf{c}, \mathbf{b}\}$  can be learned by stochastic gradient descent to minimize  $-\mathcal{L}(\theta)$  in Eq. (13.2.3), or its stochastic approximation  $-\hat{\mathcal{L}}(\theta)$  in Eq. (13.2.4), with the stochastic gradient computed by back-propagation.

Once the parameters  $\theta$  are learned, we can define a mixture model by using a uniform probability over a set of orderings  $O$ . We can compute the probability of a given vector  $\mathbf{x}$  as a mixture model

$$p_{\text{mixt}}(\mathbf{x} \mid \theta, O) = \frac{1}{|O|} \sum_{o \in O} p_{\theta}(\mathbf{x} \mid o) \quad (13.2.9)$$

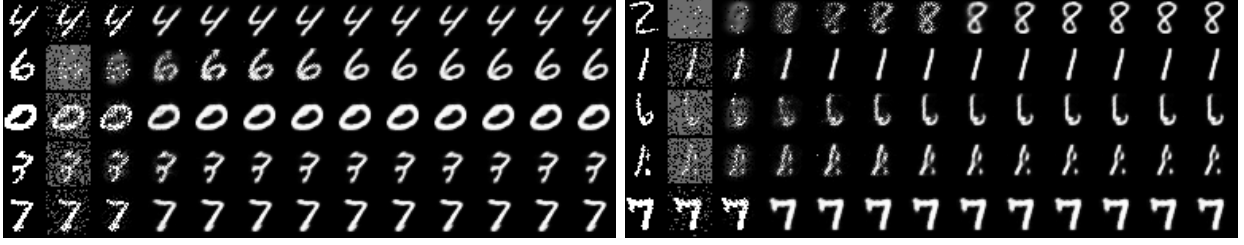


FIG. 13.2. The inner working mechanism of NADE- $k$ . The left most column shows the data vectors  $\mathbf{x}$ , the second column shows their masked version and the subsequent columns show the reconstructions  $\mathbf{v}^{(0)} \dots \mathbf{v}^{(10)}$  (See Eq. (13.2.7)).

with Eq. (13.2.2). We can draw independent samples from the mixture by first drawing an ordering  $o$  and then sequentially drawing each variable using  $x_{o_d} \sim p_{\theta}(x_{o_d} \mid \mathbf{x}_{o_{<d}})$ . Furthermore, we can draw samples from the conditional  $p(\mathbf{x}_{\text{mis}} \mid \mathbf{x}_{\text{obs}})$  easily by considering only orderings where the observed indices appear before the missing ones.

**Pretraining** It is well known that training deep networks is difficult without pretraining, and in our experiments, we train networks up to  $kn = 7 \times 3 = 21$  layers. When pretraining, we train the model to produce good reconstructions  $\mathbf{v}^{(t)}$  at each step  $t = 1 \dots k$ . More formally, in the pretraining phase, we replace Equations (13.2.4) and (13.2.8) by

$$\hat{\mathcal{L}}_{\text{pre}}(\theta) = -\frac{D}{D-d+1} \frac{1}{k} \sum_{t=1}^k \log \prod_{i \in o_{\geq d}} p_{\theta}^{(t)}(x_i \mid \mathbf{x}_{o_{<d}}) \quad (13.2.10)$$

$$p_{\theta}^{(t)}(x_i = 1 \mid \mathbf{x}_{\text{obs}}) = v_i^{(t)}. \quad (13.2.11)$$

### 13.2.1. Related Methods and Approaches

**Order-agnostic NADE** The proposed method follows closely the order-agnostic version of NADE (Uria et al., 2014), which may be considered as the special case of NADE- $k$  with  $k = 1$ . On the other hand, NADE- $k$  can be seen as a deep NADE with some specific weight sharing (matrices  $\mathbf{W}$  and  $\mathbf{V}$  are reused for different depths) and gating in the activations of some layers (See Equation (13.2.7)).

Additionally, Uria et al. (2014) found it crucial to give the mask  $\mathbf{m}$  as an auxiliary input to the network, and initialized missing values to zero instead of the empirical mean (See Eq. (13.2.5)). Due to these differences, we call their approach NADE-mask. One should note that NADE-mask has more parameters due to using the mask as a separate input to the network, whereas NADE- $k$  is roughly  $k$  times more expensive to compute.

**Probabilistic Inference** Let us consider the task of missing value imputation in a probabilistic latent variable model. We get the conditional probability of interest by marginalizing

out the latent variables from the posterior distribution:

$$p(\mathbf{x}_{\text{mis}} \mid \mathbf{x}_{\text{obs}}) = \int_{\mathbf{h}} p(\mathbf{h}, \mathbf{x}_{\text{mis}} \mid \mathbf{x}_{\text{obs}}) d\mathbf{h}. \quad (13.2.12)$$

Accessing the joint distribution  $p(\mathbf{h}, \mathbf{x}_{\text{mis}} \mid \mathbf{x}_{\text{obs}})$  directly is often harder than alternatively updating  $\mathbf{h}$  and  $\mathbf{x}_{\text{mis}}$  based on the conditional distributions  $p(\mathbf{h} \mid \mathbf{x}_{\text{mis}}, \mathbf{x}_{\text{obs}})$  and  $p(\mathbf{x}_{\text{mis}} \mid \mathbf{h})$ .<sup>1</sup> Variational inference is one of the representative examples that exploit this.

In variational inference, a factorial distribution  $q(\mathbf{h}, \mathbf{x}_{\text{mis}}) = q(\mathbf{h})q(\mathbf{x}_{\text{mis}})$  is iteratively fitted to  $p(\mathbf{h}, \mathbf{x}_{\text{mis}} \mid \mathbf{x}_{\text{obs}})$  such that the KL-divergence between  $q$  and  $p$

$$\text{KL}[q(\mathbf{h}, \mathbf{x}_{\text{mis}}) \parallel p(\mathbf{h}, \mathbf{x}_{\text{mis}} \mid \mathbf{x}_{\text{obs}})] = - \int_{\mathbf{h}, \mathbf{x}_{\text{mis}}} q(\mathbf{h}, \mathbf{x}_{\text{mis}}) \log \left[ \frac{p(\mathbf{h}, \mathbf{x}_{\text{mis}} \mid \mathbf{x}_{\text{obs}})}{q(\mathbf{h}, \mathbf{x}_{\text{mis}})} \right] d\mathbf{h} d\mathbf{x}_{\text{mis}} \quad (13.2.13)$$

is minimized. The algorithm alternates between updating  $q(\mathbf{h})$  and  $q(\mathbf{x}_{\text{mis}})$ , while considering the other one fixed.

As an example, let us consider a restricted Boltzmann machine (RBM) defined by

$$p(\mathbf{v}, \mathbf{h}) \propto \exp(\mathbf{b}^\top \mathbf{v} + \mathbf{c}^\top \mathbf{h} + \mathbf{h}^\top \mathbf{W} \mathbf{v}). \quad (13.2.14)$$

We can fit an approximate posterior distribution parameterized as  $q(v_i = 1) = \bar{v}_i$  and  $q(h_j = 1) = \bar{h}_j$  to the true posterior distribution by iteratively computing

$$\bar{\mathbf{h}} \leftarrow \sigma(\mathbf{W} \bar{\mathbf{v}} + \mathbf{c}) \quad (13.2.15)$$

$$\bar{\mathbf{v}} \leftarrow \mathbf{m} \odot \text{sigmoid}(\mathbf{W}^\top \bar{\mathbf{h}} + \mathbf{b}) + (\mathbf{1} - \mathbf{m}) \odot \mathbf{v}. \quad (13.2.16)$$

We notice the similarity to Eqs. (13.2.6)–(13.2.7): If we assume  $\phi = \text{sigmoid}$  and  $\mathbf{V} = \mathbf{W}^\top$ , the inference in the NADE- $k$  is equivalent to performing  $k$  iterations of variational inference on an RBM for the missing values (Peterson and Anderson, 1987). We can also get variational inference on a deep Boltzmann machine (DBM) using the structure in Fig. 13.1 (right).

**Multi-Prediction Deep Boltzmann Machine** Goodfellow et al. (2013) and Brakel et al. (2013) use backpropagation through variational inference steps to train a deep Boltzmann machine. This is very similar to our work, except that they approach the problem from the view of maximizing the generalized pseudo-likelihood (Huang and Ogata, 2002). Also, the deep Boltzmann machine lacks the tractable probabilistic interpretation similar to NADE- $k$  (See Eq. (13.2.2)) that would allow to compute a probability or to generate independent samples without resorting to a Markov chain. Also, our approach is somewhat more flexible in the choice of model structures, as can be seen in Fig. 13.1. For instance, in the proposed NADE- $k$ , encoding and decoding weights do not have to be shared and any type of nonlinear activations, other than a logistic sigmoid function, can be used.

---

<sup>1</sup> We make a typical assumption that observations are mutually independent given the latent variables.

**Product and Mixture of Experts** One could ask what would happen if we would define an ensemble likelihood along the line of the training criterion in Eq. (13.2.3). That is,

$$-\log p_{\text{prod}}(\mathbf{x} \mid \boldsymbol{\theta}) \propto \mathbb{E}_{o \in D} [-\log p(\mathbf{x} \mid \boldsymbol{\theta}, o)]. \quad (13.2.17)$$

Maximizing this ensemble likelihood directly will correspond to training a product-of-experts model (Hinton, 2000). However, this requires us to evaluate the intractable normalization constant during training as well as in the inference, making the model not tractable anymore.

On the other hand, we may consider using the log-probability of a sample under the mixture-of-experts model as the training criterion

$$-\log p_{\text{mixt}}(\mathbf{x} \mid \boldsymbol{\theta}) = -\log \mathbb{E}_{o \in D} [p(\mathbf{x} \mid \boldsymbol{\theta}, o)]. \quad (13.2.18)$$

This criterion resembles clustering, where individual models may specialize in only a fraction of the data. In this case, however, the simple estimator such as in Eq. (13.2.4) would not be available.

### 13.3. EXPERIMENTS

We study the proposed model with two datasets: binarized MNIST handwritten digits and Caltech 101 silhouettes.

We train NADE- $k$  with one or two hidden layers ( $n = 2$  and  $n = 3$ , see Fig. 13.1, left and middle) with a hyperbolic tangent as the activation function  $\phi(\cdot)$ . We use stochastic gradient descent on the training set with a minibatch size fixed to 100. We use AdaDelta (Zeiler, 2012) to adaptively choose a learning rate for each parameter update on-the-fly. We use the validation set for early-stopping and to select the hyperparameters. With the best model on the validation set, we report the log-probability computed on the test set. We have made our implementation available<sup>2</sup>.

#### 13.3.1. MNIST

We closely followed the procedure used by Uria et al. (2014), including the split of the dataset into 50,000 training samples, 10,000 validation samples and 10,000 test samples. We used the same version where the data has been binarized by sampling.

We used a fixed width of 500 units per hidden layer. The number of steps  $k$  was selected among  $\{1, 2, 4, 5, 7\}$ . According to our preliminary experiments, we found that no separate regularization was needed when using a single hidden layer, but in case of two hidden layers, we used weight decay with the regularization constant in the interval  $[e^{-5}, e^{-2}]$ . Each model was pretrained for 1000 epochs and fine-tuned for 1000 epochs in the case of one hidden layer and 2000 epochs in the case of two.

---

<sup>2</sup>[git@github.com:yaoli/nade\\_k.git](https://github.com:yaoli/nade_k.git)

Model	Log-Prob.	Model	Log-Prob.
NADE 1HL(fixed order)	-88.86	RBM (500h, CD-25)	$\approx$ -86.34
NADE 1HL	-99.37	DBN (500h+2000h)	$\approx$ -84.55
NADE 2HL	-95.33	DARN (500h)	$\approx$ -84.71
NADE-mask 1HL	-92.17	DARN (500h, adaNoise)	$\approx$ <b>-84.13</b>
NADE-mask 2HL	-89.17	NADE-5 1HL	-90.02
NADE-mask 4HL	-89.60	NADE-5 2HL	-87.14
EoNADE-mask 1HL(128 Ords)	-87.71	EoNADE-5 1HL(128 Ords)	-86.23
EoNADE-mask 2HL(128 Ords)	-85.10	EoNADE-5 2HL(128 Ords)	<b>-84.68</b>

TAB. 13. I. Results obtained on MNIST using various models and number of hidden layers (1HL or 2HL). “Ords” is short for “orderings”. These are the average log-probabilities of the test set. EoNADE refers to the ensemble probability (See Eq. (13.2.9)). From here on, in all figures and tables we use “HL” to denote the number of hidden layers and “h” for the number of hidden units.

For both NADE- $k$  with one and two hidden layers, the validation performance was best with  $k = 5$ . The regularization constant was chosen to be 0.00122 for the two-hidden-layer model.

**Results** We report in Table 13. I the mean of the test log-probabilities averaged over randomly selected orderings. We also show the experimental results by others from (Uria et al., 2014; Gregor et al., 2014). We denote the model proposed in (Uria et al., 2014) as a *NADE-mask*.

From Table 13. I, it is clear that NADE- $k$  outperforms the corresponding NADE-mask both with the individual orderings and ensembles over orderings using both 1 or 2 hidden layers. NADE- $k$  with two hidden layers achieved the generative performance comparable to that of the deep belief network (DBN) with two hidden layers.

Fig. 13.3 shows training curves for some of the models. We can see that the NADE-1 does not perform as well as NADE-mask. This confirms that in the case of  $k = 1$ , the auxiliary mask input is indeed useful. Also, we can note that the performance of NADE-5 is still improving at the end of the preallocated 2000 epochs, further suggesting that it may be possible to obtain a better performance simply by training longer.

Fig. 13.4 (a) shows the effect of the number of iterations  $k$  during training. Already with  $k = 2$ , we can see that the NADE- $k$  outperforms its corresponding NADE-mask. The performance increases until  $k = 5$ . We believe the worse performance of  $k = 7$  is due to the well known training difficulty of a deep neural network, considering that NADE-7 with two hidden layers effectively is a deep neural network with 21 layers.



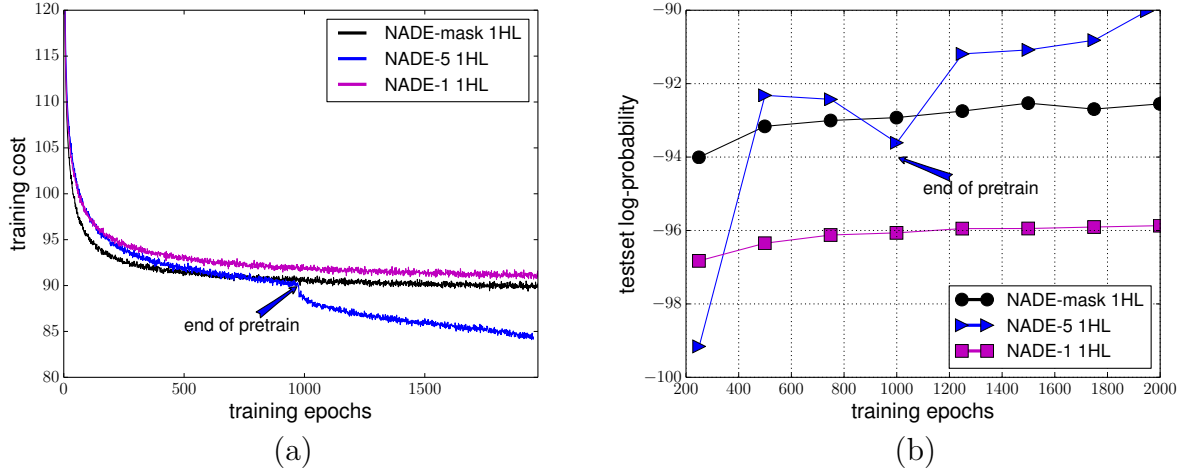


FIG. 13.3. NADE- $k$  with  $k$  steps of variational inference helps to reduce the training cost (a) and to generalize better (b). NADE-mask performs better than NADE-1 without masks both in training and test.

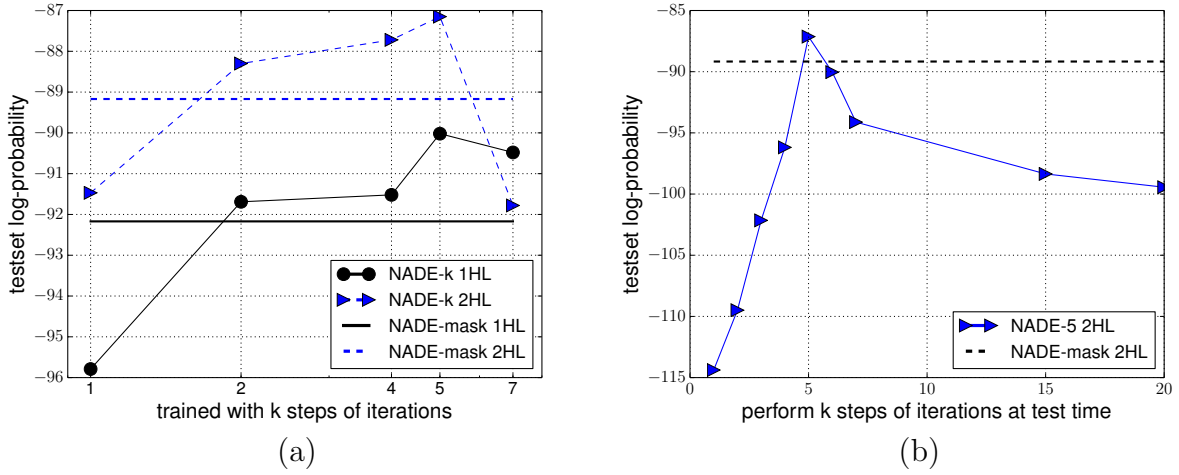


FIG. 13.4. (a) The generalization performance of different NADE- $k$  models trained with different  $k$ . (b) The generalization performance of NADE-5 2h, trained with  $k=5$ , but with various  $k$  in test time.

At inference time, we found that it is important to use the exact  $k$  that one used to train the model. As can be seen from Fig. 13.4 (b), the assigned probability increases up to the  $k$ , but starts decreasing as the number of iterations goes over the  $k$ .<sup>3</sup>

<sup>3</sup>In the future, one could explore possibilities for helping better converge beyond step  $k$ , for instance by using costs based on reconstructions at  $k - 1$  and  $k$  even in the fine-tuning phase.



### 13.3.1.1. Qualitative Analysis

In Fig. 13.2, we present how each iteration  $t = 1 \dots k$  improves the corrupted input ( $\mathbf{v}^{(t)}$  from Eq. (13.2.5)). We also investigate what happens with test-time  $k$  being larger than the training  $k = 5$ . We can see that in all cases, the iteration – which is a fixed point update – seems to converge to a point that is in most cases close to the ground-truth sample. Fig. 13.4 (b) shows however that the generalization performance drops after  $k = 5$  when training with  $k = 5$ . From Fig. 13.2, we can see that the reconstruction continues to be *sharper* even after  $k = 5$ , which seems to be the underlying reason for this phenomenon.

From the samples generated from the trained NADE-5 with two hidden layers shown in Fig. 13.5 (a), we can see that the model is able to generate digits. Furthermore, the filters learned by the model show that it has learned parts of digits such as pen strokes (See Fig. 13.6).

### 13.3.1.2. Variability over Orderings

In Section 13.2, we argued that we can perform any inference task  $p(\mathbf{x}_{\text{mis}} \mid \mathbf{x}_{\text{obs}})$  easily and efficiently by restricting the set of orderings  $O$  in Eq. (13.2.9) to ones where  $\mathbf{x}_{\text{obs}}$  is before  $\mathbf{x}_{\text{mis}}$ . For this to work well, we should investigate how much the different orderings vary.

To measure the variability over orderings, we computed the variance of  $\log p(\mathbf{x} \mid o)$  for 128 randomly chosen orderings  $o$  with the trained NADE- $k$ 's and NADE-mask with a single hidden layer. For comparison, we computed the variance of  $\log p(\mathbf{x} \mid o)$  over the 10,000 test samples.

$\log p(\mathbf{x} \mid o)$	$\mathbb{E}_{o, \mathbf{x}} [\cdot]$	$\sqrt{\mathbb{E}_{\mathbf{x}} \text{Var}_o [\cdot]}$	$\sqrt{\mathbb{E}_o \text{Var}_{\mathbf{x}} [\cdot]}$
NADE-mask 1HL	-92.17	3.5	23.5
NADE-5 1HL	-90.02	3.1	24.2
NADE-5 2HL	-87.14	2.4	22.7

TAB. 13.  
II. The  
variance of  
 $\log p(\mathbf{x} \mid o)$   
over order-  
ings  $o$  and  
over test  
samples  $\mathbf{x}$ .

In Table 13. II, the variability over the orderings is clearly much smaller than that over the samples. Furthermore, the variability over orderings tends to decrease with the better models.

### 13.3.2. Caltech-101 silhouettes

We also evaluate the proposed NADE- $k$  on Caltech-101 Silhouettes (Marlin et al., 2010), using the standard split of 4100 training samples, 2264 validation samples and 2307 test samples. We demonstrate the advantage of NADE- $k$  compared with NADE-mask under

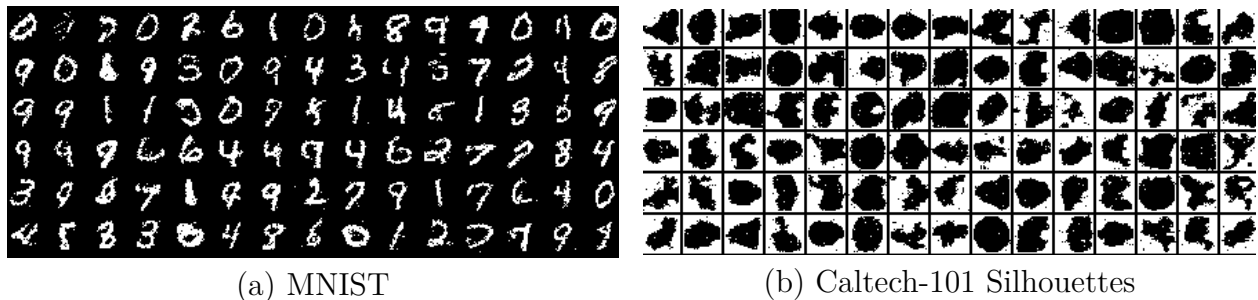


FIG. 13.5. Samples generated from NADE- $k$  trained on (a) MNIST and (b) Caltech-101 Silhouettes.

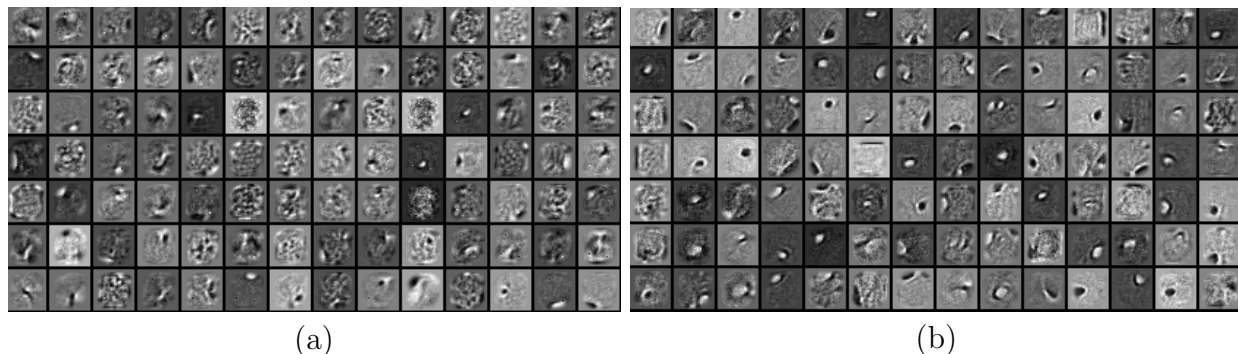


FIG. 13.6. Filters learned from NADE-5 2HL. (a) A random subset of the encoding filters. (b) A random subset of the decoding filters.

the constraint that they have a matching number of parameters. In particular, we compare NADE- $k$  with 1000 hidden units with NADE-mask with 670 hiddens. We also compare NADE- $k$  with 4000 hidden units with NADE-mask with 2670 hiddens.

We optimized the hyper-parameter  $k \in \{1, 2, \dots, 10\}$  in the case of NADE- $k$ . In both NADE- $k$  and NADE-mask, we experimented without regularizations, with weight decays, or with dropout. Unlike the previous experiments, we did not use the pretraining scheme (See Eq. (13.2.10)).

As we can see from Table 13. III, NADE- $k$  outperforms the NADE-mask regardless of the number of parameters. In addition, NADE-2 with 1000 hidden units matches the performance of an RBM with the same number of parameters. Furthermore, NADE-5 has outperformed the previous best result obtained with the RBMs in (Cho et al., 2013), achieving the state-of-art result on this dataset. We can see from the samples generated by the NADE- $k$  shown in Fig. 13.5 (b) that the model has learned the data well.

## 13.4. CONCLUSIONS AND DISCUSSION

In this paper, we proposed a model called iterative neural autoregressive distribution estimator (NADE- $k$ ) that extends the conventional neural autoregressive distribution estimator (NADE) and its order-agnostic training procedure. The proposed NADE- $k$  maintains the

TAB. 13. III. Average log-probabilities of test samples of Caltech-101 Silhouettes. (★) The results are from [Cho et al. \(2013\)](#). The terms in the parenthesis indicate the number of hidden units, the total number of parameters (M for million), and the L2 regularization coefficient. NADE-mask 670h achieves the best performance without any regularizations.

Model	Test LL	Model	Test LL
RBM★ (2000h, 1.57M)	-108.98	RBM ★ (4000h, 3.14M)	<b>-107.78</b>
NADE-mask (670h, 1.58M)	-112.51	NADE-mask (2670h, 6.28M, L2=0.00106)	-110.95
NADE-2 (1000h, 1.57M, L2=0.0054)	-108.81	NADE-5 (4000h, 6.28M, L2=0.0068)	<b>-107.28</b>

tractability of the original NADE while we showed that it outperforms the original NADE as well as similar, but intractable generative models such as restricted Boltzmann machines and deep belief networks.

The proposed extension is inspired from the variational inference in probabilistic models such as restricted Boltzmann machines (RBM) and deep Boltzmann machines (DBM). Just like an iterative mean-field approximation in Boltzmann machines, the proposed NADE- $k$  performs multiple iterations through hidden layers and a visible layer to infer the probability of the missing value, unlike the original NADE which performs the inference of a missing value in a single iteration through hidden layers.

Our empirical results show that this approach of multiple iterations improves the performance of a model that has the same number of parameters, compared to performing a single iteration. This suggests that the inference method has significant effect on the efficiency of utilizing the model parameters. Also, we were able to observe that the generative performance of NADE can come close to more sophisticated models such as deep belief networks in our approach.

In the future, more in-depth analysis of the proposed NADE- $k$  is needed. For instance, a relationship between NADE- $k$  and the related models such as the RBM need to be both theoretically and empirically studied. The computational speed of the method could be improved both in training (by using better optimization algorithms. See, e.g., ([Pascanu and Bengio, 2014](#))) and in testing (e.g. by handling the components in chunks rather than fully sequentially). The computational efficiency of sampling for NADE- $k$  can be further improved based on the recent work of [Yao et al. \(2014\)](#) where an annealed Markov chain may be used to efficiently generate samples from the trained ensemble. Another promising idea to improve the model performance further is to let the model adjust its own confidence based on  $d$ . For instance, in the top right corner of Fig. 13.2, we see a case with lots of

missing values values (low  $d$ ), where the model is too confident about the reconstructed digit 8 instead of the correct digit 2.

### *Acknowledgements*

The authors would like to acknowledge the support of NSERC, Calcul Québec, Compute Canada, the Canada Research Chair and CIFAR, and developers of Theano ([Bergstra et al., 2010](#); [Bastien et al., 2012](#)).

# Chapter 14

---

## RECENT DEVELOPMENT IN LEARNING VISUAL REPRESENTATIONS

### 14.1. LEARNING IMAGE REPRESENTATIONS

The success of supervised learning hinges on the availability of a large labelled dataset. When such a condition is not satisfied, one resorts to the use of unsupervised learning to obtain image representations. The idea of training a deep and directed generative model have prevailed over the years. This includes the work in Chapter 11 and Chapter 13. Generative Adversarial Networks (GAN) introduced in Goodfellow et al. (2014) suggests training such type of models with a minmax optimization procedure. This clever approach avoids the difficulty of maximizing the log-likelihood directly on pixels but instead utilizes a classifier to differentiate real and synthesized images. A close relative to GAN is Generative Moment Matching proposed in (Li et al., 2015) where the Maximum Mean Discrepancy (Gretton et al., 2012) is used in place of the standard minmax criteria. Recently, (Arjovsky and Bottou, 2017) provides theoretical analysis of the problems including instability and saturation that arise when training GANs.

On the other hand, Variational Autoencoders (VAEs) (see Section 2.6.2) proposed in Kingma and Welling (2013) have remained an active research area through the years with several noticeable contributions. Gregor et al. (2015) drastically improved the generative performance on MNIST, a commonly used benchmark for generative models since 2006. The work of Kingma et al. (2016) suggests the use a non-factorized distribution to approximate the true posterior instead of the commonly used diagonal Gaussian. Theis et al. (2017); Ballé et al. (2016) use VAEs to compress images. One particular interesting work is Higgins et al. (2017) where VAEs are used as the model for learning early visual concept, highlighting a connection to the findings in neuroscience.

Unlike VAEs where samples are generated entirely at once, NADEs, including both the deep and shallow ones mentioned in Chapter 11 and Chapter 13 rely on a predefined ordering when generating samples. Such a sequential aspect is the direct result of the chain rule of

factorization of the joint probability of inputs. Recently, instead of parameterizing it with NADEs, PixelCNNs (Oord et al., 2016; Salimans et al., 2017) parameterize it with LSTMs (section 2.4) that models the details of images well. The current state-of-the-art performance on binarized MNIST (a popular benchmark among generative models) is claimed by Gulrajani et al. (2017) where VAEs and PixelCNNs are combined together to learn a meaningful latent representations while capturing large structures in images.

Unlike VAEs where the log-probability of a deep directed generative models (such as VAEs) is typically intractable due to the marginalization of latent variables. Nonlinear Independent Component Estimations (NICEs) from Dinh et al. (2014, 2017) utilize a bijective encoder that results a model that enables exact and tractable log-likelihood, inference and sampling.

Models described above, including GANs, VAEs, NADEs, PixelRNNs and NICEs have their pros and cons. For instance, GANs are able to generate by far sharpest images but do not explicitly define a likelihood function and training such models could also be challenging due to the unstable training dynamics. VAEs have formally defined likelihood function and are efficient in generating samples but they tend to be slightly blurry. PixelRNNs and NADEs have tractable likelihood function but the generation is inefficient. NICEs have both a tractable likelihood function and an efficient generating process, yet the capacity of such models are limited. The future will undoubtedly see development in generative models that are both statistically and computationally more efficient.

## 14.2. LEARNING VIDEO REPRESENTATIONS

The work of Chapter 5, Chapter 7 and Chapter 9 belongs to the exciting field of learning video representations. There are many ways to achieve this. The dominant approach by far has been the use of supervised learning. Action recognition is perhaps the most representative and the most studied in this category. Despite its dominance in image applications, neural networks have yet to enjoy the equal success in video applications. The reasons are manyfolded. Firstly, a labelled video dataset is hard to obtain. Among the largest ones the number of examples range from tens of thousands (Soomro et al., 2012; Kuehne et al., 2013; Fabian Caba Heilbron and Nibbles, 2015) to hundreds of thousands (Torabi et al., 2015; Rohrbach et al., 2015b). To this end, there has been significant effort in the community recently to collect large datasets with good quality. The most noticeable is Abu-El-Haija et al. (2016) where 8 million YouTube videos are collected and labelled for the general purpose of video understanding.

A good representation can be learned with a direct supervised signal. But convolutional and recurrent neural networks that are so popular in handling 2D signal become computationally prohibitive as one example of a short video typically contains 50 to 100 frames. The issue of insufficient training examples only compounds the problem. This motivates

the work in Chapter 5 where a 3DConvNet is trained from local statistics instead of pixels (such as in Tran et al. (2015) and Karpathy et al. (2014)) to reduce the computational complexity. Another popular and successful alternative is to adopt a two-stream approach as in Wang et al. (2015); Ye et al. (2015); Simonyan and Zisserman (2014a); Feichtenhofer et al. (2016) where one 2DConvNet is trained on single frames to capture spatial patterns and the other one trained on pre-computed Optical Flows to explicitly model temporal correlations. This family of model currently claims the state-of-the-art in action recognition among all learning-based approaches.

A good representation may also be learned with an unsupervised signal. Srivastava et al. (2015a) predicts past and future frames given the current ones to learned representations that capture both spatial and temporal correlations. Such an LSTM model is improved in Chapter 9 where recurrent neural networks are made convolutional. Similarly Oh et al. (2015) use deconvolution to predict next frames in Atari games. Goroshin et al. (2015) encourage temporal coherence in addition to making frame predictions. Mathieu et al. (2016) make the important observation that the commonly used mean square error in frame prediction is prone to produce blurry results, and propose alternative training criteria to sharpen it. Instead of predicting frames, Finn et al. (2016) explicitly models the transformation between frames in a confined environment. The most recent work for frame prediction is Lotter et al. (2017) where the prediction error is fed back into the model for correction, inspired by findings in neuroscience. Villegas et al. (2017) also predicts future frames but proposes to decompose motion and content. Bazzani et al. (2017) combines both 3DConvNets and LSTMs with visual attention (similar to those used in Chapter 5) to predict visual saliency, and the learned representation improves action recognition on UCF101. Temporal coherence can be further exploited by predicting the ordering such as the work in Misra et al. (2016).





# Chapter 15

---

## CONCLUSION

It has been nearly 5 years since the revolutionary work of [Krizhevsky et al. \(2012b\)](#) where a deep neural network was trained with supervision on fast GPUs. It has been a decade since the dawn of deep learning in 2006 ([Hinton and Salakhutdinov, 2006b](#)) where a deep autoencoder was trained layer by layer with restricted Boltzmann machines. And it has been nearly 75 years since the first mathematical model of a neuron was proposed in [McCulloch and Pitts \(1943b\)](#). Nowadays we have developed computers that are capable of accomplishing amazing things such as identifying objects, driving vehicles and playing games. None of this would have been possible without representing visual percepts in a mathematical format readable by computers. There has been ample evidence that the most efficient way of obtaining such a representation is through learning.

This work contributes in this direction. In particular, the second half of this thesis is devoted to studying models that learn to represent images with unsupervised learning. The first half is devoted to studying those that learn to represent videos with supervised learning. While supervised neural networks have matured over the years, unsupervised models have advanced relatively slowly as it is much more challenging to build unsupervised models that are statistical and computational efficient, and such a grand challenge arguably aligns with our understanding of the working mechanism of the brain.

The future will likely to see great scientific advancement in unsupervised learning in both images and videos. This could be achieved by both a deeper understanding of visual perception in the brain and an efficient computational model that mimics it. Faster hardware, such as GPUs being used today is certainly an indispensable part of a computational revolution that is yet to come.



# Bibliography

---

- Abu-El-Haija, S., N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan (2016). Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*.
- Alain, G. and Y. Bengio (2013). What regularized auto-encoders learn from the data generating distribution. In *International Conference on Learning Representations (ICLR'2013)*.
- Anandan, P. (1989). A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision* 2(3), 283–310.
- Antol, S., A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh (2015). Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2425–2433.
- Arjovsky, M. and L. Bottou (2017). Towards principled methods for training generative adversarial networks. In *ICLR*, Volume 2016.
- Bahdanau, D., K. Cho, and Y. Bengio (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Bahdanau, D., K. Cho, and Y. Bengio (2015). Neural machine translation by jointly learning to align and translate. *ICLR*.
- Baldi, P. and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima.
- Baldi, P. and K. Hornik (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks* 2(1), 53–58.
- Ballas, N., L. Yao, C. Pal, and A. Courville (2016). Delving deeper into convolutional networks for learning video representations. *ICLR*.
- Ballé, J., V. Laparra, and E. P. Simoncelli (2016). End-to-end optimized image compression. *ICLR*.
- Barbu, A., A. Bridge, Z. Burchill, D. Coroian, S. Dickinson, S. Fidler, A. Michaux, S. Mussman, S. Narayanaswamy, D. Salvi, et al. (2012). Video in sentences out. *UAI*.
- Bastien, F., P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio (2012). Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

- Bazzani, L., H. Larochelle, and L. Torresani (2017). Recurrent mixture density network for spatiotemporal visual attention.
- Bengio, S., O. Vinyals, N. Jaitly, and N. Shazeer (2015). Scheduled sampling for sequence prediction with recurrent neural networks. *arXiv preprint arXiv:1506.03099*.
- Bengio, Y. (2013). Deep learning of representations: looking forward. In *Statistical Language and Speech Processing*, Volume 7978 of *Lecture Notes in Computer Science*, pp. 1–37. Springer.
- Bengio, Y. et al. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning* 2(1), 1–127.
- Bengio, Y. and S. Bengio (2000). Modeling high-dimensional discrete data with multi-layer neural networks. pp. 400–406. MIT Press.
- Bengio, Y., A. Courville, and P. Vincent (2013). Representation learning: A review and new perspectives. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)* 35(8), 1798–1828.
- Bengio, Y., O. Delalleau, and N. Le Roux (2006). The curse of highly variable functions for local kernel machines. In Y. Weiss, B. Schölkopf, and J. Platt (Eds.), *Advances in Neural Information Processing Systems 18*, pp. 107–114. Cambridge, MA: MIT Press.
- Bengio, Y., O. Delalleau, and C. Simard (2010a). Decision trees do not generalize to new variations. *Computational Intelligence* 26(4), 449–467.
- Bengio, Y., O. Delalleau, and C. Simard (2010b, nov). Decision trees do not generalize to new variations. *Computational Intelligence* 26(4), 449–467.
- Bengio, Y., R. Ducharme, P. Vincent, and C. Janvin (2003). A neural probabilistic language model. *The Journal of Machine Learning Research* 3, 1137–1155.
- Bengio, Y., P. Lamblin, D. Popovici, and H. Larochelle (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pp. 153–160.
- Bengio, Y., E. Laufer, G. Alain, and J. Yosinski (2014). Deep generative stochastic networks trainable by backprop. In *Proceedings of The 31st International Conference on Machine Learning*, pp. 226–234.
- Bengio, Y. and Y. LeCun (2007). Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston (Eds.), *Large Scale Kernel Machines*. MIT Press.
- Bengio, Y., D.-H. Lee, J. Bornschein, T. Mesnard, and Z. Lin (2015). Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*.
- Bengio, Y., G. Mesnil, Y. Dauphin, and S. Rifai (2013). Better mixing via deep representations.
- Bengio, Y., P. Simard, and P. Frasconi (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5(2), 157–166.
- Bengio, Y., E. Thibodeau-Laufer, G. Alain, and J. Yosinski (2014). Deep generative stochastic networks trainable by backprop. Technical Report arXiv:1306.1091.

- Bengio, Y., L. Yao, G. Alain, and P. Vincent (2013a). Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pp. 899–907.
- Bengio, Y., L. Yao, G. Alain, and P. Vincent (2013b). Generalized denoising auto-encoders as generative models. In *NIPS’2013*.
- Bergstra, J. and Y. Bengio (2012). Random search for hyper-parameter optimization. *JMLR*.
- Bergstra, J., O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*.
- Besag, J. (1975). Statistical analysis of non-lattice data. *The Statistician* 24(3), 179–195.
- Bishop, C. M. (1994). Mixture density networks.
- Boden, M. A. (2006). *Mind as machine: A history of cognitive science*. Clarendon Press.
- Bojanowski, P., R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic (2014). Weakly supervised action labeling in videos under ordering constraints. In *ECCV*.
- Bottou, L. (2013). Large-scale learning with stochastic gradient descent. In K.-R. Müller, G. Montavon, and G. B. Orr (Eds.), *Neural Networks: Tricks of the Trade, Reloaded*. Springer.
- Brakel, P., D. Stroobandt, and B. Schrauwen (2013). Training energy-based models for time-series imputation. *The Journal of Machine Learning Research* 14(1), 2771–2797.
- Brox, T. and J. Malik (2011). Large displacement optical flow: descriptor matching in variational motion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*.
- Chen, D. L. and W. B. Dolan (2011a). Collecting highly parallel data for paraphrase evaluation. In *ACL*.
- Chen, D. L. and W. B. Dolan (2011b). Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 190–200. Association for Computational Linguistics.
- Chen, X., H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollar, and C. L. Zitnick (2015). Microsoft coco captions: Data collection and evaluation server. *arXiv 1504.00325*.
- Cho, K., T. Raiko, and A. Ilin (2013). Enhanced gradient for training restricted boltzmann machines. *Neural computation* 25(3), 805–831.
- Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.
- Cho, K., B. van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio (2014, October). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

- Choromanska, A., M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun (2015). The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pp. 192–204.
- Chung, J., C. Gulcehre, K. Cho, and Y. Bengio (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
- Dalal, N. and B. Triggs (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Volume 1, pp. 886–893. IEEE.
- Dalal, N., B. Triggs, and C. Schmid (2006). Human detection using oriented histograms of flow and appearance. In *ECCV*.
- Dauphin, Y. N., R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pp. 2933–2941.
- Dayan, P. and L. Abbott (2003). Theoretical neuroscience: computational and mathematical modeling of neural systems. *Journal of Cognitive Neuroscience* 15(1), 154–155.
- Denkowski, M. and A. Lavie (2014). Meteor universal: Language specific translation evaluation for any target language. In *EACL Workshop*.
- Devlin, J., H. Cheng, H. Fang, S. Gupta, L. Deng, X. He, G. Zweig, and M. Mitchell (2015). Language models for image captioning: The quirks and what works. *arXiv preprint arXiv:1505.01809*.
- Devlin, J., S. Gupta, R. Girshick, M. Mitchell, and C. L. Zitnick (2015). Exploring nearest neighbor approaches for image captioning. *arXiv preprint arXiv:1505.04467*.
- Dinh, L., D. Krueger, and Y. Bengio (2014). Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.
- Dinh, L., J. Sohl-Dickstein, and S. Bengio (2017). Density estimation using real nvp.
- Domke, J. (2011). Parameter learning with truncated message-passing. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 2937–2943. IEEE.
- Donahue, J., L. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell (2014). Long-term recurrent convolutional networks for visual recognition and description. *arXiv preprint arXiv:1411.4389*.
- Fabian Caba Heilbron, Victor Escorcia, B. G. and J. C. Niebles (2015). Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–970.
- Fang, H., S. Gupta, F. Iandola, R. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. Platt, et al. (2015). From captions to visual concepts and back. *CVPR*.
- Farabet, C., C. Couprie, L. Najman, and Y. LeCun (2013). Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence* 35(8), 1915–1929.

- Fei-Fei, L. and P. Perona (2005). A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Volume 2, pp. 524–531. IEEE.
- Feichtenhofer, C., A. Pinz, and A. Zisserman (2016). Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1933–1941.
- Finn, C., I. Goodfellow, and S. Levine (2016). Unsupervised learning for physical interaction through video prediction. In *Advances In Neural Information Processing Systems*, pp. 64–72.
- Forsyth, D. A. and J. Ponce (2003). A modern approach. *Computer vision: a modern approach*, 88–101.
- Gaidon, A., Z. Harchaoui, and C. Schmid (2013). Temporal localization of actions with actoms. *PAMI*.
- Geyer, C. J. (1992). Practical markov chain monte carlo. *Statistical Science*, 473–483.
- Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448.
- Girshick, R., J. Donahue, T. Darrell, and J. Malik (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587.
- Glorot, X. and Y. Bengio (2010). Understanding the difficulty of training deep feedforward neural networks. In *AISTATS’2010*.
- Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- Goodfellow, I., M. Miraz, A. Courville, and Y. Bengio (2013, December). Multi-prediction deep Boltzmann machines. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger (Eds.), *Advances in Neural Information Processing Systems 26*, pp. 548–556.
- Goodfellow, I., M. Mirza, A. Courville, and Y. Bengio (2013). Multi-prediction deep boltzmann machines. In *Advances in Neural Information Processing Systems*, pp. 548–556.
- Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680.
- Goroshin, R., J. Bruna, J. Tompson, D. Eigen, and Y. LeCun (2015). Unsupervised learning of spatiotemporally coherent metrics. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4086–4093.
- Graves, A. and N. Jaitly (2014). Towards end-to-end speech recognition with recurrent neural networks. In *ICML*.

- Gregor, K., I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra (2015). Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*.
- Gregor, K., I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra (2014). Deep autoregressive networks. In *International Conference on Machine Learning (ICML'2014)*.
- Gretton, A., K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola (2012). A kernel two-sample test. *Journal of Machine Learning Research* 13(Mar), 723–773.
- Grosse, R. and J. Martens (2016). A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pp. 573–582.
- Guadarrama, S., N. Krishnamoorthy, G. Malkarnenkar, S. Venugopalan, R. Mooney, T. Darrell, and K. Saenko (2013). Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *ICCV*.
- Gulrajani, I., K. Kumar, F. Ahmed, A. A. Taiga, F. Visin, D. Vazquez, and A. Courville (2017). Pixelvae: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*.
- Hawkins, J. and S. Blakeslee (2004, October). *On Intelligence* (Adapted ed.). Times Books.
- He, K., X. Zhang, S. Ren, and J. Sun (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.
- Heckerman, D., D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie (2000). Dependency networks for inference, collaborative filtering, and data visualization. 1, 49–75.
- Higgins, I., L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
- Hinton, G. (2000). Training products of experts by minimizing contrastive divergence. Technical Report GCNU TR 2000-004, Gatsby Unit, University College London.
- Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Amherst 1986, pp. 1–12. Lawrence Erlbaum, Hillsdale.
- Hinton, G. E., S. Osindero, and Y.-W. Teh (2006a). A fast learning algorithm for deep belief nets. *Neural computation* 18(7), 1527–1554.
- Hinton, G. E., S. Osindero, and Y. W. Teh (2006b). A fast learning algorithm for deep belief nets. *Neural Computation* 18, 1527–1554.
- Hinton, G. E. and R. Salakhutdinov (2006a, July). Reducing the dimensionality of data with neural networks. *Science* 313(5786), 504–507.
- Hinton, G. E. and R. R. Salakhutdinov (2006b). Reducing the dimensionality of data with neural networks. *science* 313(5786), 504–507.



- Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2012a). Improving neural networks by preventing co-adaptation of feature detectors. Technical report, arXiv:1207.0580.
- Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov (2012b). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.
- Hochreiter, S. and J. Schmidhuber (1997a). Long short-term memory. *Neural computation*.
- Hochreiter, S. and J. Schmidhuber (1997b). Long short-term memory. *Neural Computation*.
- Hodosh, M., P. Young, and J. Hockenmaier (2013). Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*.
- Horn, B. K. and B. G. Schunck (1981). Determining optical flow. *Artificial intelligence* 17(1-3), 185–203.
- Huang, F. and Y. Ogata (2002). Generalized pseudo-likelihood estimates for Markov random fields on lattice. *Annals of the Institute of Statistical Mathematics* 54(1), 1–18.
- Hyvärinen, A. (2007). Some extensions of score matching. *Computational Statistics and Data Analysis* 51, 2499–2512.
- Hyvärinen, A. and E. Oja (2000). Independent component analysis: algorithms and applications. *Neural networks* 13(4), 411–430.
- Jégou, H., M. Douze, C. Schmid, and P. Pérez (2010). Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 3304–3311. IEEE.
- Ji, S., W. Xu, M. Yang, and K. Yu (2013). 3d convolutional neural networks for human action recognition. *PAMI*.
- Jia, X., E. Gavves, B. Fernando, and T. Tuytelaars (2015). Guiding long-short term memory for image caption generation. *arXiv preprint arXiv:1509.04942*.
- Jia, Y., E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*.
- Jiang, Y., J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar (2014). Thumos challenge: Action recognition with a large number of classes. *Technical Report*.
- Kalchbrenner, N., A. v. d. Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu (2016). Video pixel networks. *arXiv preprint arXiv:1610.00527*.
- Karhunen, J., E. Oja, L. Wang, R. Vigario, and J. Joutsensalo (1997). A class of neural networks for independent component analysis. *IEEE Transactions on Neural Networks* 8(3), 486–504.
- Karpathy, A. and L. Fei-Fei (2014). Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

- Karpathy, A., G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei (2014). Large-scale video classification with convolutional neural networks. In *CVPR*. IEEE.
- Kingma, D. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kingma, D. P., T. Salimans, and M. Welling (2016). Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*.
- Kingma, D. P. and M. Welling (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
- Kiros, R., R. Salakhutdinov, and R. S. Zemel (2014a). Unifying visual-semantic embeddings with multimodal neural language models. *ACL*.
- Kiros, R., R. Salakhutdinov, and R. S. Zemel (2014b). Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.
- Klaser, A., M. Marszałek, and C. Schmid (2008). A spatio-temporal descriptor based on 3d-gradients. In *BMVC 2008-19th British Machine Vision Conference*, pp. 275–1. British Machine Vision Association.
- Kojima, A., T. Tamura, and K. Fukunaga (2002). Natural language description of human activities from video images based on concept hierarchy of actions. *IJCV*.
- Krizhevsky, A., I. Sutskever, and G. Hinton (2012a). ImageNet classification with deep convolutional neural networks. In *NIPS’2012*.
- Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012b). Imagenet classification with deep convolutional neural networks. In *NIPS*.
- Kuehne, H., H. Jhuang, R. Stiefelhagen, and T. Serre (2013). Hmdb51: A large video database for human motion recognition. In *High Performance Computing in Science and Engineering ’12*, pp. 571–582. Springer.
- Kulkarni, G., V. Premraj, V. Ordonez, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg (2013). Babytalk: Understanding and generating simple image descriptions. *PAMI*.
- Kuznetsova, P., V. Ordonez, A. C. Berg, T. L. Berg, and Y. Choi (2012). Collective generation of natural image descriptions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pp. 359–368. Association for Computational Linguistics.
- Lan, Z., M. Lin, X. Li, A. G. Hauptmann, and B. Raj (2014). Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. *arXiv preprint arXiv:1411.6660*.
- Laptev, I. (2005). On space-time interest points. *International journal of computer vision* 64(2-3), 107–123.
- Larochelle, H. and I. Murray (2011a). The Neural Autoregressive Distribution Estimator. In *AISTATS’2011*.
- Larochelle, H. and I. Murray (2011b). The neural autoregressive distribution estimator. *Journal of Machine Learning Research* 15, 29–37.

- LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *Nature* 521(7553), 436–444.
- LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.
- LeCun, Y. A., L. Bottou, G. B. Orr, and K.-R. Müller (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer.
- Li, Y., K. Swersky, and R. Zemel (2015). Generative moment matching networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1718–1727.
- Lin, T.-Y., M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick (2014). Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014*, pp. 740–755. Springer.
- Long, J., E. Shelhamer, and T. Darrell (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440.
- Lotter, W., G. Kreiman, and D. Cox (2017). Deep predictive coding networks for video prediction and unsupervised learning.
- Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, Volume 2, pp. 1150–1157. Ieee.
- Lucas, B. D., T. Kanade, et al. (1981). An iterative image registration technique with an application to stereo vision.
- Lyu, S. (2009). Interpretation and generalization of score matching.
- Mao, J., W. Xu, Y. Yang, J. Wang, and A. Yuille (2015). Deep captioning with multimodal recurrent neural networks (m-rnn). *ICLR*.
- Marlin, B., K. Swersky, B. Chen, and N. de Freitas (2010). Inductive principles for restricted Boltzmann machine learning. Volume 9, pp. 509–516.
- Martens, J. (2010, June). Deep learning via Hessian-free optimization. pp. 735–742.
- Martens, J. and R. Grosse (2015). Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pp. 2408–2417.
- Martens, J. and I. Sutskever (2011). Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1033–1040.
- Mathieu, M., C. Couprie, and Y. LeCun (2016). Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*.
- McCulloch, W. S. and W. Pitts (1943a). A logical calculus of ideas immanent in nervous activity. 5, 115–133.
- McCulloch, W. S. and W. Pitts (1943b). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5(4), 115–133.

- Milletari, F., N. Navab, and S.-A. Ahmadi (2016). V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3D Vision (3DV), 2016 Fourth International Conference on*, pp. 565–571. IEEE.
- Misra, I., C. L. Zitnick, and M. Hebert (2016). Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pp. 527–544. Springer.
- Mitchell, M., X. Han, J. Dodge, A. Mensch, A. Goyal, A. Berg, K. Yamaguchi, T. Berg, K. Stratos, and H. Daumé III (2012). Midge: Generating image descriptions from computer vision detections. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 747–756. Association for Computational Linguistics.
- Morsillo, N., G. Mann, and C. Pal (2010). Youtube scale, large vocabulary video annotation. In *Video Search and Mining*.
- Neal, R. M. (1994). Sampling from multimodal distributions using tempered transitions. Technical Report 9421, Dept. of Statistics, University of Toronto.
- Neal, R. M. (2001, April). Annealed importance sampling. *Statistics and Computing* 11(2), 125–139.
- Neal, R. M. and G. E. Hinton (1998). A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pp. 355–368. Springer.
- Ng, J. Y.-H., M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici (2015). Beyond short snippets: Deep networks for video classification. *arXiv preprint arXiv:1503.08909*.
- Oh, J., X. Guo, H. Lee, R. L. Lewis, and S. Singh (2015). Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pp. 2863–2871.
- Oja, E. (1992). Principal components, minor components, and linear neural networks. *Neural networks* 5(6), 927–935.
- Olshausen, B. A. and D. J. Field (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* 381(6583), 607.
- Oord, A. v. d., N. Kalchbrenner, and K. Kavukcuoglu (2016). Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*.
- Pan, P., Z. Xu, Y. Yang, F. Wu, and Y. Zhuang (2015). Hierarchical recurrent neural encoder for video representation with application to captioning. *arXiv preprint arXiv:1511.03476*.
- Papineni, K., S. Roukos, T. Ward, and W.-J. Zhu (2002). Bleu: a method for automatic evaluation of machine translation. In *ACL*.
- Pascanu, R. and Y. Bengio (2014). Revisiting natural gradient for deep networks.
- Pascanu, R., T. Mikolov, and Y. Bengio (2013). On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pp. 1310–1318.

- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
- Perronnin, F., J. Sánchez, and T. Mensink (2010). Improving the fisher kernel for large-scale image classification. *Computer Vision–ECCV 2010*, 143–156.
- Peterson, C. and J. R. Anderson (1987). A mean field theory learning algorithm for neural networks. *Complex Systems* 1(5), 995–1019.
- Prince, S. J. (2012). *Computer vision: models, learning, and inference*. Cambridge University Press.
- Qi Wu, Q., C. Shen, A. van den Hengel, L. Liu, and A. Dick (2015). What value high level concepts in vision to language problems? *arXiv 1506.01144*.
- Radford, A., L. Metz, and S. Chintala (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.
- Ranzato, M., Y.-L. Boureau, and Y. LeCun (2007). Sparse feature learning for deep belief networks. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pp. 1185–1192. Curran Associates Inc.
- Ranzato, M., A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra (2014). Video (language) modeling: a baseline for generative models of natural videos. *arXiv: 1412.6604*.
- Reed, S., Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee (2016). Generative adversarial text to image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*, Volume 3.
- Ren, S., K. He, R. Girshick, and J. Sun (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99.
- Riesenhuber, M. and T. Poggio (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*.
- Robbins, H. and S. Monro (1951). A stochastic approximation method. *Annals of Mathematical Statistics* 22, 400–407.
- Rohrbach, A., M. Rohrbach, and B. Schiele (2015). The long-short story of movie description.
- Rohrbach, A., M. Rohrbach, N. Tandon, and B. Schiele (2015a). A dataset for movie description. *CVPR*.
- Rohrbach, A., M. Rohrbach, N. Tandon, and B. Schiele (2015b). A dataset for movie description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Rohrbach, M., W. Qiu, I. Titov, S. Thater, M. Pinkal, and B. Schiele (2013). Translating video content to natural language descriptions. In *ICCV*.
- Ronneberger, O., P. Fischer, and T. Brox (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241. Springer.

- Rosenblatt, F. (1957). The perceptron — a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, N.Y.
- Sadanand, S. and J. Corso (2012). Action bank: A high-level representation of activity in video. In *CVPR*. IEEE.
- Salakhutdinov, R. and G. Hinton (2009a). Deep boltzmann machines. In *Artificial Intelligence and Statistics*, pp. 448–455.
- Salakhutdinov, R. and G. Hinton (2009b). Deep Boltzmann machines. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS 2009)*, Volume 8.
- Salakhutdinov, R. and G. Hinton (2009c). Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, Volume 5, pp. 448–455.
- Salimans, T., A. Karpathy, X. Chen, and D. P. Kingma (2017). Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*.
- Saul, L. K., T. Jaakkola, and M. I. Jordan (1996). Mean field theory for sigmoid belief networks. *Journal of artificial intelligence research* 4(1), 61–76.
- Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks* 61, 85–117.
- Scovanner, P., S. Ali, and M. Shah (2007). A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th ACM international conference on Multimedia*, pp. 357–360. ACM.
- Sermanet, P., D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.
- Sermanet, P., D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. *ICLR*.
- Shi, X., Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214*.
- Siegelmann, H. T. and E. D. Sontag (1995). On the computational power of neural nets. *Journal of computer and system sciences* 50(1), 132–150.
- Simonyan, K. and A. Zisserman (2014a). Two-stream convolutional networks for action recognition in videos. *NIPS*.
- Simonyan, K. and A. Zisserman (2014b). Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pp. 568–576.
- Simonyan, K. and A. Zisserman (2014c). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

- Simonyan, K. and A. Zisserman (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR*.
- Sohl-Dickstein, J., P. Battaglino, and M. R. DeWeese (2009). Minimum probability flow learning. Technical report, arXiv:0906.4779.
- Soomro, K., A. R. Zamir, and M. Shah (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.
- Srivastava, N., E. Mansimov, and R. Salakhutdinov (2015a). Unsupervised learning of video representations using lstms. *arXiv: 1502.04681*.
- Srivastava, N., E. Mansimov, and R. Salakhutdinov (2015b). Unsupervised learning of video representations using lstms. In *ICML*.
- Sterratt, D., B. Graham, A. Gillies, and D. Willshaw (2011). *Principles of computational modelling in neuroscience*. Cambridge University Press.
- Stinchcombe, M. and H. White (1989). Universal approximation using feedforward networks with non-sigmoid hidden layer activation function. Washington DC, pp. 613–617. IEEE.
- Stoyanov, V., A. Ropson, and J. Eisner (2011). Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *International Conference on Artificial Intelligence and Statistics*, pp. 725–733.
- Sun, D., S. Roth, and M. J. Black (2014). A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision* 106(2), 115–137.
- Sutskever, I. and G. E. Hinton (2008). Deep, narrow sigmoid belief networks are universal approximators. *Neural Computation* 20(11), 2629–2636.
- Sutskever, I., O. Vinyals, and Q. V. V. Le (2014). Sequence to sequence learning with neural networks. In *NIPS*.
- Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich (2014). Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*.
- Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich (2015). Going deeper with convolutions. *CVPR*.
- Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.
- Tang, K., L. Fei-Fei, and D. Koller (2012). Learning latent temporal structure for complex event detection. In *CVPR*. IEEE.
- Taylor, G. W., R. Fergus, Y. LeCun, and C. Bregler (2010). Convolutional learning of spatio-temporal features. In *Computer Vision–ECCV 2010*, pp. 140–153. Springer.
- Theano Development Team (2016, May). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints abs/1605.02688*.
- Theis, L., W. Shi, A. Cunningham, and F. Huszár (2017). Lossy image compression with compressive autoencoders.

- Thomason, J., S. Venugopalan, S. Guadarrama, K. Saenko, and R. Mooney (2014). Integrating language and vision to generate natural language descriptions of videos in the wild. In *COLING*.
- Tieleman, T. (2008). Training restricted boltzmann machines using approximations to the likelihood gradient. pp. 1064–1071.
- Torabi, A., C. Pal, H. Larochelle, and A. Courville (2015). Using descriptive video services to create a large data source for video annotation research. *arXiv: 1503.01070*.
- Tran, D., L. Bourdev, R. Fergus, L. Torresani, and M. Paluri (2014). C3d: generic features for video analysis. *arXiv preprint arXiv:1412.0767*.
- Tran, D., L. Bourdev, R. Fergus, L. Torresani, and M. Paluri (2015). Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4489–4497.
- Trappenberg, T. (2009). *Fundamentals of computational neuroscience*. OUP Oxford.
- Tsai, C.-F. (2012). Bag-of-words representation in image annotation: A review. *ISRN Artificial Intelligence 2012*.
- Uria, B., I. Murray, and H. Larochelle (2013a). A deep and tractable density estimator. Technical Report arXiv:1310.1757.
- Uria, B., I. Murray, and H. Larochelle (2013b). Rnade: The real-valued neural autoregressive density-estimator. In *NIPS’2013*.
- Uria, B., I. Murray, and H. Larochelle (2014). A deep and tractable density estimator. In *Proceedings of the 30th International Conference on Machine Learning (ICML’14)*.
- Vedantam, R., C. L. Zitnick, and D. Parikh (2014). CIDEr: Consensus-based image description evaluation. *arXiv:1411.5726*.
- Venugopalan, S., M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko (2015). Sequence to sequence – video to text. In *ICCV*.
- Venugopalan, S., H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko (2015). Translating videos to natural language using deep recurrent neural networks. *NAACL*.
- Villegas, R., J. Yang, S. Hong, X. Lin, and H. Lee (2017). Decomposing motion and content for natural video sequence prediction.
- Vincent, P., H. Larochelle, Y. Bengio, and P.-A. Manzagol (2008a). Extracting and composing robust features with denoising autoencoders. In *Int. Conf. Mach. Learn.*, pp. 1096–1103.
- Vincent, P., H. Larochelle, Y. Bengio, and P.-A. Manzagol (2008b). Extracting and composing robust features with denoising autoencoders. In *ICML 2008*.
- Vincent, P., H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Machine Learning Res. 11*.



- Vinyals, O., A. Toshev, S. Bengio, and D. Erhan (2014). Show and tell: A neural image caption generator. *CVPR*.
- Viola, P. and M. Jones (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, Volume 1, pp. I–I. IEEE.
- Vondrick, C., H. Pirsiavash, and A. Torralba (2016). Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pp. 613–621.
- Wang, H., A. Kläser, C. Schmid, and L. Cheng-Lin (2011). Action recognition by jectories. In *CVPR*.
- Wang, H., A. Kläser, C. Schmid, and C. Liu (2011). Action recognition by dense trajectories. In *CVPR*. IEEE.
- Wang, H. and C. Schmid (2013). Action recognition with improved trajectories. In *ICCV*.
- Wang, H., M. M. Ullah, A. Kläser, I. Laptev, and C. Schmid (2009). Evaluation of local spatio-temporal features for action recognition. In *University of Central Florida, U.S.A.*
- Wang, J., J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong (2010). Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 3360–3367. IEEE.
- Wang, L., Y. Qiao, and X. Tang (2015). Action recognition with trajectory-pooled deep-convolutional descriptors. *arXiv preprint arXiv:1505.04868*.
- Wang, L., Y. Xiong, Z. Wang, and Y. Qiao (2015). Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*.
- Weston, J., F. Ratle, and R. Collobert (2008). Deep learning via semi-supervised embedding. In *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML 2008)*.
- Widrow, B. and M. Lehr (1990, September). 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation. *Proceedings of the IEEE* 78(9), 1415–1442.
- Willems, G., T. Tuytelaars, and L. Van Gool (2008). An efficient dense and scale-invariant spatio-temporal interest point detector. *Computer Vision–ECCV 2008*, 650–663.
- Wolpert, D. H. (1996). The lack of a priori distinction between learning algorithms. *Neural Computation* 8(7), 1341–1390.
- Xu, H., S. Venugopalan, V. Ramanishka, M. Rohrbach, and K. Saenko (2015). A multi-scale multiple instance video description network. *arXiv 1505.05914*.
- Xu, K., J. Ba, R. Kiros, , K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio (2015). Show, attend and tell: Neural image caption generation with visual attention. *ICML*.
- Yang, J., K. Yu, Y. Gong, and T. Huang (2009). Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1794–1801. IEEE.

- Yao, L., N. Ballas, K. Cho, J. R. Smith, and Y. Bengio (2015). Trainable performance upper bounds for image and video captioning. *arXiv preprint arXiv:1511.0459*.
- Yao, L., S. Ozair, K. Cho, and Y. Bengio (2014). On the equivalence between deep nade and generative stochastic networks. In *European Conference on Machine Learning (ECML/PKDD’14)*. Springer.
- Yao, L., A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville (2015a). Describing videos by exploiting temporal structure. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pp. 4507–4515.
- Yao, L., A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville (2015b). Describing videos by exploiting temporal structure. In *ICCV*.
- Ye, H., Z. Wu, R.-W. Zhao, X. Wang, Y.-G. Jiang, and X. Xue (2015). Evaluating two-stream cnn for video classification. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pp. 435–442. ACM.
- Young, P., A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *ACL14*.
- Yu, H., J. Wang, Z. Huang, Y. Yang, and W. Xu (2015). Video paragraph captioning using hierarchical recurrent neural networks. *arXiv 1510.07712*.
- Zaremba, W., I. Sutskever, and O. Vinyals (2014). Recurrent neural network regularization. *arXiv: 1409.2329*.
- Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. Technical report.
- Zhou, B., A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva (2014). Learning deep features for scene recognition using places database. In *NIPS14*.